

COLT 2010

The 23rd Conference on Learning Theory

**Haifa, Israel
June 27-29, 2010**

Edited by Adam Tauman Kalai and Mehryar Mohri

Cover was designed by Andrea Victoria Peñaredondo

Copyright © 2010
These materials are owned by their respective copyright owners.

All rights reserved.

ISBN number: 978-0-9822529-2-5

Forward

COLT 2010, the 23rd International Conference on Learning Theory, took place from June 27-29 in Haifa, Israel, at the Dan Carmel Hotel. There was a day trip to Jerusalem and a banquet in Nazareth. The conference was co-located with ICML 2010, and the two shared a joint workshop day. Forty-one papers were accepted out of 129 submissions. The submission deadline was February 19, 2010. Six open problems were accepted out of nine submissions, with a deadline of March 13th. The best paper prize was awarded to Ohad Shamir, Shai Shalev-Shwartz and Karthik Sridharan, for “Learning Kernel-Based Halfspaces with the Zero-One Loss” and the best student paper prize was awarded to Purnamrita Sarkar for “Theoretical Justification of Popular Link Prediction Heuristics” (joint with Deepayan Chakrabarti and Andrew Moore). The local organizers were Shai Fine and Oded Margalit, from IBM Research Haifa. The Volunteer Chair was Noam Slonim, publicity chair was Sandra Zilles, and the Open Problems Chair was Claudio Gentile. The program committee was:

- Shivani Agarwal (Massachusetts Institute of Technology)
- Mikhail Belkin (Ohio State University)
- Shai Ben-David (University of Waterloo)
- Nicol Cesa-Bianchi (University of Milan)
- Ofer Dekel (Microsoft)
- Steve Hanneke (Carnegie Mellon University)
- Jeff Jackson (Duquesne University)
- Adam Tauman Kalai (Microsoft)
- Sham Kakade (University of Pennsylvania)
- Vladimir Koltchinskii (Georgia Institute of Technology)
- Katrina Ligett (Cornell University)
- Phil Long (Google)
- Gabor Lugosi (Pompeu Fabra University)
- Ulrike von Luxburg (Max Planck Institute)
- Yishay Mansour (Tel-Aviv University and Google)
- Mehryar Mohri (Courant Institute and Google Research)
- Ryan O'Donnell (Carnegie Mellon University)
- Massimiliano Pontil (University College London)
- Robert Schapire (Princeton University)
- Rocco Servedio (Columbia University)
- Shai Shalev-Shwartz (Hebrew University)
- John Shawe-Taylor (University College London)
- Gilles Stoltz (Ecole Normale Suprieure)
- Ambuj Tewari (Toyota Technological Institute)
- Jenn Wortman Vaughan (Harvard University)
- Santosh Vempala (Georgia Institute of Technology)
- Manfred Warmuth (UC Santa Cruz)
- Robert Williamson (Australian National University)
- Thomas Zeugmann (Hokkaido University)
- Tong Zhang (Rutgers University)

The conference was unanimously hailed a great success.

Sincerely,

Adam Tauman Kalai and Mehryar Mohri, the program chairs

Colt 2010 Sponsors

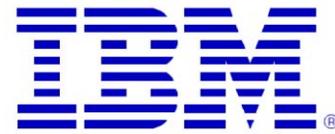


Table of Contents

Convex Games in Banach Spaces	1
<i>Karthik Sridharan, Ambuj Tewari</i>	
Composite Objective Mirror Descent	14
<i>John C. Duchi, Shai Shalev-Shwartz, Yoram Singer, Ambuj Tewari</i>	
Voting Paradoxes	27
<i>Noga Alon</i>	
Optimal Algorithms for Online Convex Optimization with Multi-Point Bandit Feedback	28
<i>Alekh Agarwal, Ofer Dekel, Lin Xiao</i>	
Best Arm Identification in Multi-Armed Bandits	41
<i>Jean-Yves Audibert, Sebastien Bubeck, Remi Munos</i>	
Nonparametric Bandits with Covariates	54
<i>Philippe Rigollet, Assaf Zeevi</i>	
An Asymptotically Optimal Bandit Algorithm for Bounded Support Models	67
<i>Junya Honda, Akimichi Takemura</i>	
Learning with Global Cost in Stochastic Environments	80
<i>Eyal Even-dar, Shie Mannor, Yishay Mansour</i>	
Hedging Structured Concepts	93
<i>Wouter M. Koolen, Manfred K. Warmuth, Jyrki Kivinen</i>	
Following the Flattened Leader	106
<i>Wojciech Kotlowski, Peter Grunwald, Steven de Rooij</i>	
Sequence prediction in realizable and non-realizable cases	119
<i>Daniil Ryabko</i>	
Regret Minimization for Online Bounding Problems Using the Weighted Majority Algorithm	132
<i>Sascha Geulen, Berthold Vöcking, Melanie Winkler</i>	
Learning rotations with little regret	144
<i>Elad Hazan, Satyen Kale, Manfred K. Warmuth</i>	
Evolution with Drifting Targets	155
<i>Varun Kanade, Leslie G. Valiant, Jennifer Wortman Vaughan</i>	
Regret Minimization With Concept Drift	168
<i>Koby Crammer, Yishay Mansour, Eyal Even-Dar, Jennifer Wortman Vaughan</i>	
Strongly Non-U-Shaped Learning Results by General Techniques	181
<i>John Case, Timo Kötzing</i>	
Inferring Descriptive Generalisations of Formal Languages	194
<i>Dominik D. Freydenberger, Daniel Reidenbach</i>	
Quantum Predictive Learning and Communication Complexity with Single Input	207
<i>Dmitry Gavinsky</i>	
Online Learning of Noisy Data with Kernels	218
<i>Nicola Cesa-Bianchi, Shai Shalev Shwartz, Ohad Shamir</i>	
The Online Loop-free Stochastic Shortest-Path Problem	231
<i>Gergely Neu, Andras Gyorgy, Csaba Szepesvari</i>	
Adaptive Bound Optimization for Online Convex Optimization	244
<i>H. Brendan McMahan, Matthew Streeter</i>	
Adaptive Subgradient Methods for Online Learning and Stochastic Optimization	257
<i>John Duchi, Elad Hazan, Yoram Singer</i>	

Table of Contents

Characterization of Linkage-based Clustering	270
<i>Margareta Ackerman, Shai Ben-David, David Loker</i>	
Robust Hierarchical Clustering	282
<i>Maria Florina Balcan, Pramod Gupta</i>	
Theoretical Justification of Popular Link Prediction Heuristics	295
<i>Purnamrita Sarkar, Deepayan Chakrabarti, Andrew W. Moore</i>	
The Convergence Rate of AdaBoost	308
<i>Robert E. Schapire</i>	
Learning Talagrand DNF Formulas	310
<i>Homin K. Lee</i>	
Open Problem: Analyzing Ant Robot Coverage	312
<i>Sven Koenig</i>	
On-line variance minimization in $O(n^2)$ per trial?	314
<i>Elad Hazan, Satyen Kale, Manfred K. Warmuth</i>	
Robust Efficient Conditional Probability Estimation	316
<i>John Langford</i>	
Can We Learn to Gamble Efficiently?	318
<i>Jacob Abernethy</i>	
Active Learning on Trees and Graphs	320
<i>Nicolo Cesa-Bianchi, Claudio Gentile, Fabio Vitale, Giovanni Zappella</i>	
Adaptive Submodularity: A New Approach to Active Learning and Stochastic Optimization	333
<i>Daniel Golovin, Andreas Krause</i>	
Robust Selective Sampling from Single and Multiple Teachers	346
<i>Ofer Dekel, Claudio Gentile, Karthik Sridharan</i>	
Improved Guarantees for Agnostic Learning of Disjunctions	359
<i>Pranjal Awasthi, Avrim Blum, Or Sheffet</i>	
Mansour's Conjecture is True for Random DNF Formulas	368
<i>Adam Klivans, Homin K. Lee, Andrew Wan</i>	
Deterministic Sparse Fourier Approximation via Fooling Arithmetic Progressions	381
<i>Adi Akavia</i>	
Forest Density Estimation	394
<i>Anupam Gupta, John Lafferty, Han Liu, Larry Wasserman, Min Xu</i>	
Toward Learning Gaussian Mixtures with Arbitrary Separation	407
<i>Mikhail Belkin, Kaushik Sinha</i>	
Sparse Recovery in Convex Hulls of Infinite Dictionaries	420
<i>Vladimir Koltchinskii, Stas Minsker</i>	
Efficient classification for metric data	433
<i>Lee-Ad Gottlieb, Aryeh (Leonid) Kontorovich, Robert Krauthgamer</i>	
Learning Kernel-Based Halfspaces with the Zero-One Loss	441
<i>Shai Shalev-Shwartz, Ohad Shamir, Karthik Sridharan</i>	
Ranking with kernels in Fourier space	451
<i>Risi Kondor, Marconi Barbosa</i>	
Causal Markov condition for submodular information measures	464
<i>Bastian Steudel, Dominik Janzing, Bernhard Scholkopf</i>	

Table of Contents

Open Loop Optimistic Planning	477
<i>Sebastien Bubeck, Remi Munos</i>	
Principal Component Analysis with Contaminated Data: The High Dimensional Case	490
<i>Huan Xu, Constantine Caramanis, Shie Mannor</i>	
Robustness and Generalization	503
<i>Huan Xu, Shie Mannor</i>	
Learning to Create is as Hard as Learning to Appreciate	516
<i>David Xiao</i>	

Convex Games in Banach Spaces

Karthik Sridharan
TTI-Chicago
karthik@ttic.edu

Ambuj Tewari
TTI-Chicago
tewari@ttic.edu

Abstract

We study the regret of an online learner playing a multi-round game in a Banach space \mathfrak{B} against an adversary that plays a convex function at each round. We characterize the minimax regret when the adversary plays linear functions in terms of the Martingale type of the dual of \mathfrak{B} . The cases when the adversary plays bounded and uniformly convex functions respectively are also considered. Our results connect online convex learning to the study of the geometry of Banach spaces. We also show that appropriate modifications of the Mirror Descent algorithm from convex optimization can be used to achieve our regret upper bounds. Finally, we provide a version of Mirror Descent that adapts to the changing exponent of uniform convexity of the adversary's functions. This adaptive mirror descent strategy provides new algorithms even for the more familiar Hilbert space case where the loss functions on each round have varying exponents of uniform convexity (curvature).

1 Introduction

Online convex optimization [1, 2, 3] has emerged as an abstraction that allows a unified treatment of a variety of online learning problems where the underlying loss function is convex. In this abstraction, a T -round game is played between the learner (or the player) and an adversary. At each round $t \in \{1, \dots, T\}$, the player makes a move \mathbf{w}_t in some set \mathcal{W} . In the learning context, the set \mathcal{W} will represent some hypothesis space. Once the player has made his choice, the adversary then picks a *convex* function ℓ_t from some set \mathcal{F} and the player suffers "loss" $\ell_t(\mathbf{w}_t)$. In the learning context, the adversary's move ℓ_t encodes the data seen at time t and the loss function used to measure the performance of \mathbf{w}_t on that data. As with any abstraction, on one hand, we lose contact with the concrete details of the problem at hand, but on the other hand, we gain the ability to study related problems from a unified point of view. An added benefit of this abstraction is that it connects online learning with geometry of convex sets, theory of optimization and game theory.

An important notion in the online setting is that of the cumulative *regret* incurred by the player which is the difference between the cumulative loss of the player and the cumulative loss for the best fixed move in hindsight. The goal of regret minimizing algorithms is to control the growth rate of the regret as a function of T . There has been a huge amount of work characterizing the best regret rates possible under a variety of assumptions on the player's and adversary's sets \mathcal{W} and \mathcal{F} . With a few exceptions that we mention later, most of the work has been in the setting where these sets live in some Euclidean space \mathbb{R}^d . Whenever the results do not explicitly involve the dimensionality d , they are also usually applicable in any Hilbert space \mathfrak{H} . There also has been a lot of work dealing with ℓ_p spaces for $p \in [1, 2]$. But here, the fact exploited is that a strongly convex function, (with dimension-independent constant of strong convexity) is available in these Banach spaces. There has been less work dealing with arbitrary Banach spaces (where strongly convex functions might not exist). Our focus in this paper is to extend the study of optimal regret rates to the case when the set \mathcal{W} lives in a general Banach space \mathfrak{B} .

Before we explain what our specific contributions are, let us briefly mention two examples to show why one might want to move beyond Hilbert spaces and consider general Banach spaces. A first family of examples is ℓ_p spaces with $p \geq 2$. As p grows, the ℓ_p balls become larger and thus have better approximation properties. On the other hand, as we show below, the cumulative regret rate when competing against a fixed size ball in ℓ_p is $O(T^{1/p})$. So, there is a trade-off here between

approximation properties and the regret rate (or the estimation error in a stochastic setting). In high-dimensions, one can easily construct examples where the approximation property dominates the trade-off and it is advantageous to use $p > 2$ even though the estimation error suffers (see appendix for a worked out example). Another example is prediction with squared loss where the learner is trying to predict a signal y_t given input x_t . At each step, the learner chooses a *function* f_t and suffers the loss $(y_t - f_t(x_t))^2$. Here, the viewpoint of considering f_t as a “point” in a function space is very fruitful and it very natural to assume that the space of functions that the learner can use is a Banach space of functions. For more details, see [4].

In the Hilbert space setting, it is known that the “degree of convexity” or “curvature” of the functions ℓ_t played by the adversary has a significant impact on the achievable regret rates. For example, if the adversary can play arbitrary convex and Lipschitz functions, the best regret possible is $O(\sqrt{T})$. However, if the adversary is constrained to play *strongly convex* and Lipschitz functions, the regret can be brought down to $O(\log T)$. Further, it is also known, via minimax lower bounds [5], that these are the best possible rates in these situations. In a general Banach space, strongly convex functions might not even exist. We will, therefore, need a generalization of strong convexity called *q-uniform convexity* (strong convexity is 2-uniform convexity). There will, in general, be a number $q^* \in [2, \infty)$ such that q^* -uniformly convex functions are the “most curved” functions available on \mathfrak{B} . There are, again, two extremes: the adversary can play either arbitrary convex-Lipschitz functions or q^* -uniformly convex functions. We show that the minimax optimal rates in these two situations are of the order $\Theta^*(T^{1/p^*})$ and $\Theta^*(T^{2-q^*})$ respectively¹ where p^* is the Martingale type of the dual \mathfrak{B}^* of \mathfrak{B} . A Hilbert space has $p^* = q^* = 2$. We also give upper and lower bounds for the intermediate case when the adversary plays q -uniformly convex functions for $q > q^*$. This case, as far as we know, has not been analyzed *even in the Hilbert space setting*.

Another natural game that we have not seen analyzed before is the convex-bounded game: here the adversary plays convex and bounded functions. Of course, being Lipschitz on a bounded domain implies boundedness but the reverse implication is false: a bounded function can have arbitrarily bad Lipschitz constant. For the convex-bounded game, we do not have a tight characterization but we can give non-trivial upper bounds. However, these upper bounds suffice to prove, for example, that the following three properties of \mathfrak{B} are equivalent: (1) the convex-bounded game when the player plays in the unit ball of \mathfrak{B} has non-trivial (i.e. $o(T)$) minimax regret; (2) the corresponding convex-Lipschitz game has non-trivial minimax regret; and (3) the Banach space \mathfrak{B} is super-reflexive.

We further describe player strategies that achieve the optimal rates for these convex games. These strategies are all based on the *Mirror Descent* algorithm that originated in the convex optimization literature [6]. Usually Mirror Descent is run with a strongly convex function but it turns out that it can also be analyzed in our Banach space setting if it is run with a q -uniformly convex function Ψ . Moreover, with the correct choice of Ψ , it achieves all the upper bounds presented in this paper. Thus, part of our contribution is also to show the remarkable properties of the Mirror Descent algorithm.

Our final contribution is an adaptive algorithm, building on previous work, that *adapts* to the exponent of uniform convexity in the adversary’s functions. Our results have novel implications even in a Hilbert space. For example, [7] showed how to adapt to an adversary that mixes linear and strong convex functions in its moves. We can now allow this mix to also consist of functions with intermediate degrees of uniform convexity.

Related work The idea of exploiting minimax-maximin duality to analyze optimal regret rates also appears in the recent work of Abernethy et al. [8]. The earliest papers we know of that explore the connection of the type of a Banach space to learning theory are those of Donahue et al. [9] and Gurvits [10]. Mendelson and Schechtman [11] gave estimates of the fat-shattering dimension of linear functionals on a Banach space in terms of its type. In the context of online regression with squared loss, Vovk [4] also gives rates worse than $O(\sqrt{T})$ when the class of functions one is competing against is not in a Hilbert space, but in some Banach space. He also mentions “Banach Learning” as an open problem in his *online prediction wiki*². For recent work exploring Banach spaces for learning applications, see [12, 13, 14]. These papers also give more reasons for considering general Banach spaces in Learning Theory.

Outline The rest of the paper is organized as follows. In Section 2, we formally define the minimax and maximin values of the game between a player and an adversary. We also introduce the notions of Martingale type and uniform convexity from functional analysis. Section 3 considers convex-Lipschitz and linear games. A key result in that section is Theorem 4 which gives a characterization

¹Our informal $\Theta^*(\cdot)$ notation hides factors that are $o(T^\epsilon)$ for every $\epsilon > 0$.

²<http://onlineprediction.net/?n=Open.BanachLearning>

of the minimax regret of these games in terms of the type. Convex-bounded games are treated in Section 4 and the equivalence of super-reflexivity of the space to the existence of player strategies achieving non-trivial regret guarantees is established (Corollary 6). We next consider the case when the adversary plays “curved” functions in Section 5. Here the regret depends on the exponent of uniform convexity of the functions played by the adversary (Theorem 8). In Section 6, we describe player strategies based on the Mirror Descent algorithm that achieve the upper bounds presented in Section 3 and Section 4. In Section 7, using the techniques of Bartlett et al. [7], we give a player strategy that adapts to the exponent of uniform convexity of the functions being played by the adversary. We conclude in Section 8 by exploring directions for future work including a discussion on how the ideas in this paper might lead to practical algorithms. All proofs omitted from the main body of the paper can be found in the appendix.

2 Preliminaries

2.1 Regret and Minimax Value

Our primary objects of study are certain T -round games where the player \mathcal{P} makes moves in convex set \mathcal{W} contained in some (real) separable Banach space \mathfrak{B} . The adversary \mathcal{A} plays bounded continuous convex functions on \mathcal{W} chosen from a fixed function class \mathcal{F} . The game proceeds as follows.

For $t = 1$ to T

- \mathcal{P} plays $\mathbf{w}_t \in \mathcal{W}$,
- \mathcal{A} plays $\ell_t \in \mathcal{F}$,
- \mathcal{P} suffers $\ell_t(\mathbf{w}_t)$.

For given sequences $\mathbf{w}_{1:T}, \ell_{1:T}$, we define the *regret* of \mathcal{P} as,

$$\text{Reg}(\mathbf{w}_{1:T}, \ell_{1:T}) := \sum_{t=1}^T \ell_t(\mathbf{w}_t) - \inf_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^T \ell_t(\mathbf{w}) .$$

Given the tuple $(T, \mathfrak{B}, \mathcal{W}, \mathcal{F})$, we can define the minimax value of the above game as follows.

Definition 1. Given $T \geq 1$ and $\mathfrak{B}, \mathcal{W}, \mathcal{F}$ satisfying the conditions above, define the minimax value,

$$V_{T, \mathfrak{B}}(\mathcal{W}, \mathcal{F}) := \inf_{\mathbf{w}_1 \in \mathcal{W}} \sup_{\ell_1 \in \mathcal{F}} \cdots \inf_{\mathbf{w}_T \in \mathcal{W}} \sup_{\ell_T \in \mathcal{F}} \text{Reg}(\mathbf{w}_{1:T}, \ell_{1:T}) .$$

When T and \mathfrak{B} are clear from context, we will simply denote the minimax value by $V(\mathcal{W}, \mathcal{F})$. A *player strategy* (or \mathcal{P} -strategy) W is a sequence (W_1, \dots, W_T) of functions such that $W_t : \mathcal{F}^{t-1} \rightarrow \mathcal{W}$. For a strategy W , we define the regret as,

$$\text{Reg}(W, \ell_{1:T}) := \sum_{t=1}^T \ell_t(W_t(\ell_{1:t-1})) - \inf_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^T \ell_t(\mathbf{w}) .$$

In terms of player strategies, the minimax value takes a simpler form,

$$V_{T, \mathfrak{B}}(\mathcal{W}, \mathcal{F}) = \inf_W \sup_{\ell_{1:T}} \text{Reg}(W, \ell_{1:T}) ,$$

where the supremum is over all sequences $\ell_{1:T} \in \mathcal{F}^T$. Let Q denote distributions over \mathcal{F}^T . We can define the maximin value,

$$U_{T, \mathfrak{B}}(\mathcal{W}, \mathcal{F}) := \sup_Q \inf_W \mathbb{E}_{\ell_{1:T} \sim Q} [\text{Reg}(W, \ell_{1:T})] .$$

One has that

$$V_{T, \mathfrak{B}}(\mathcal{W}, \mathcal{F}) \geq U_{T, \mathfrak{B}}(\mathcal{W}, \mathcal{F}) . \tag{1}$$

This inequality will be the starting point for our subsequent analysis. For more about the minimax and maximin values refer, [15, p. 31].

2.2 Martingale type and Uniform Convexity

One of the goals of this paper is to characterize the minimax value in terms of the geometric properties of the space \mathfrak{B} and “degree of convexity” inherent in the functions in \mathcal{F} . Among the geometric characteristics of a Banach space \mathfrak{B} , the most useful for us is the notion of *Martingale type* (or *M-type*) of \mathfrak{B} . A Banach space \mathfrak{B} has *M-type* p if there is some constant C such that for any $T \geq 1$ and martingale difference sequence $\mathbf{d}_1, \dots, \mathbf{d}_T$ with values in \mathfrak{B} ,

$$\mathbb{E} \left[\left\| \sum_{t=1}^T \mathbf{d}_t \right\|^2 \right] \leq C \left(\sum_{t=1}^T \mathbb{E} [\|\mathbf{d}_t\|^p] \right)^{1/p} . \tag{2}$$

We also define the best M-type possible for a Banach space,

$$p^*(\mathfrak{B}) := \sup \{p : \mathfrak{B} \text{ has M-type } p\} . \quad (3)$$

A Banach space \mathfrak{B} has *M-cotype* q if there is some constant C such that for any $T \geq 1$ and martingale difference sequence $\mathbf{d}_1, \dots, \mathbf{d}_T$ with values in \mathfrak{B} ,

$$\left(\sum_{t=1}^T \mathbb{E} [\|\mathbf{d}_t\|^q] \right)^{1/q} \leq C \mathbb{E} \left[\left\| \sum_{t=1}^T \mathbf{d}_t \right\| \right] . \quad (4)$$

A closely related notion in Banach space theory is that of super-reflexivity. Refer [16] for more details.

Definition 2. A Banach space \mathfrak{B} is *super-reflexive* if no non-reflexive space is finitely representable in \mathfrak{B} .

A result of Pisier [16] shows that a Banach space \mathfrak{B} has non-trivial M-type ($p^* > 1$) (or equivalently non-trivial Martingale co-type) if and only if it is super-reflexive.

To measure the “degree of convexity” of the functions played by the adversary, we need the notion of uniform convexity. Let $\|\cdot\|$ be the norm associated with a Banach space \mathfrak{B} . A function $\ell : \mathfrak{B} \rightarrow \mathbb{R}$ is said to be (C, q) -uniformly convex on \mathfrak{B} if there is some constant $C > 0$ such that, for any $\mathbf{v}_1, \mathbf{v}_2 \in \mathfrak{B}$ and any $\theta \in [0, 1]$,

$$\ell(\theta \mathbf{v}_1 + (1 - \theta) \mathbf{v}_2) \leq \theta \ell(\mathbf{v}_1) + (1 - \theta) \ell(\mathbf{v}_2) - \frac{C\theta(1 - \theta)}{q} \|\mathbf{v}_1 - \mathbf{v}_2\|^q .$$

If $C \geq 1$ we simply say that the function ℓ is q -uniformly convex.

The following remarkable theorem of Pisier [17] shows that the concept of M-types and existence of uniformly convex functions in the Banach space are intimately connected.

Theorem (Pisier). A Banach space \mathfrak{B} has M-cotype q iff there exists a q -uniformly convex function on \mathfrak{B} .

Now consider some $p \in [1, p_M^*(\mathfrak{B}^*)]$. Then, by definition, \mathfrak{B}^* has M-type p . It is a fact that \mathfrak{B}^* has M-type p iff \mathfrak{B} has M-cotype $\frac{p}{p-1}$ [16, Chapter 6]. Thus, \mathfrak{B} has M-cotype $\frac{p}{p-1}$. Now, Pisier’s theorem guarantees the existence of a $\frac{p}{p-1}$ -uniformly convex function on \mathfrak{B} .

For a convex function $\ell : \mathfrak{B} \rightarrow \mathbb{R}$, its *subdifferential* at a point \mathbf{v} is defined as,

$$\partial \ell(\mathbf{v}) = \{ \boldsymbol{\lambda} \in \mathfrak{B}^* : \forall \mathbf{v}', \ell(\mathbf{v}') \geq \ell(\mathbf{v}) + \boldsymbol{\lambda}(\mathbf{v}' - \mathbf{v}) \} ,$$

where \mathfrak{B}^* denotes the *dual space* of \mathfrak{B} . This consists of all continuous linear functions on \mathfrak{B} with norm defined as $\|\ell\|_* := \sup_{\mathbf{w}: \|\mathbf{w}\| \leq 1} \ell(\mathbf{w})$. If $\partial \ell(\mathbf{v})$ is a singleton then we say ℓ is differentiable at \mathbf{v} and denote the unique member of $\partial \ell(\mathbf{v})$ by $\nabla \ell(\mathbf{v})$. If ℓ is differentiable at \mathbf{v}_1 , define the *Bregman divergence* associated with ℓ as,

$$\Delta_\ell(\mathbf{v}_1, \mathbf{v}_2) = \ell(\mathbf{v}_1) - \ell(\mathbf{v}_2) - \nabla \ell(\mathbf{v}_2)(\mathbf{v}_1 - \mathbf{v}_2) .$$

Recall that a function $\ell : \mathfrak{B} \rightarrow \mathbb{R}$ is L -Lipschitz on some set $\mathcal{W} \in \mathfrak{B}$ if for any $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{W}$, we have $\ell(\mathbf{v}_1) - \ell(\mathbf{v}_2) \leq L \|\mathbf{v}_1 - \mathbf{v}_2\|$. Given a set \mathcal{W} in a Banach space \mathfrak{B} , we define the following natural sets of convex functions on \mathcal{W} ,

$$\begin{aligned} \text{lin}(\mathcal{W}) &:= \{ \ell : \ell \text{ is linear and 1-Lipschitz on } \mathcal{W} \} , \\ \text{cvx}(\mathcal{W}) &:= \{ \ell : \ell \text{ is convex and 1-Lipschitz on } \mathcal{W} \} , \\ \text{bdd}(\mathcal{W}) &:= \{ \ell : \ell \text{ is convex and bounded by 1 on } \mathcal{W} \} , \\ \text{cvx}_{q,L}(\mathcal{W}) &:= \{ \ell : \ell \text{ is } q\text{-uniformly convex and } L\text{-Lipschitz on } \mathcal{W} \} . \end{aligned}$$

In the following sections, we will analyze the minimax value $V(\mathcal{W}, \mathcal{F})$ when the adversary’s set of moves is one these 4 sets defined above. For readability, we will drop the dependence of these sets on \mathcal{W} when it is clear from context. For example, we will refer to $V(\mathcal{W}, \text{cvx}(\mathcal{W}))$ simply as $V(\mathcal{W}, \text{cvx})$.

3 Convex-Lipschitz and Linear Games

Given a Banach space \mathfrak{B} with a norm $\|\cdot\|$, denote its unit ball by $U(\mathfrak{B}) := \{\mathbf{v} \in \mathfrak{B} : \|\mathbf{v}\| \leq 1\}$. Consider the case when the \mathcal{P} 's set \mathcal{W} is the unit ball $U(\mathfrak{B})$ for some \mathfrak{B} . This setting is not as restrictive as it sounds since any bounded *symmetric* convex set K in a vector space \mathbf{V} gives a Banach space $\mathfrak{B} = (\mathbf{V}, \|\cdot\|_K)$, where we equip \mathbf{V} with the norm,

$$\|\mathbf{v}\|_K := \inf \{\alpha > 0 : \mathbf{v} \in \alpha K\} . \quad (5)$$

Moreover, (the closure of) K is the unit ball of this Banach space.

So, fix \mathfrak{B} and consider the case $\mathcal{W} = U(\mathfrak{B})$, $\mathcal{F} = \text{cvx}(\mathcal{W})$. Theorem 14 given in [5] gives us $V(\mathcal{W}, \text{cvx}) = V(\mathcal{W}, \text{lin})$. We are therefore led to consider the case $\mathcal{W} = U(\mathfrak{B})$, $\mathcal{F} = \text{lin}(\mathcal{W})$. Note that $\text{lin}(\mathcal{W})$ is simply the unit ball $U(\mathfrak{B}^*)$. The theorem below relates the minimax value $V(U(\mathfrak{B}), \text{lin})$ to the behaviour of martingale difference sequences in \mathfrak{B}^* .

Theorem 3. *The minimax value $V(U(\mathfrak{B}), \text{lin})$ of the linear game is bounded as,*

$$V(U(\mathfrak{B}), \text{lin}) \geq \sup_{\mathbf{M}} \mathbb{E} \left[\left\| \sum_{t=1}^T \ell_t \right\|_{\star} \right] .$$

where the supremum is over distributions \mathbf{M} of martingale difference $(\ell_t)_{t=1}^T$ such that each $\ell_t \in U(\mathfrak{B}^*)$.

Proof. Recall that we denote a general distribution over \mathcal{A} 's sequences by Q and \mathcal{P} -strategies by W . Equation (1) gives us,

$$V(\mathcal{W}, \mathcal{F}) \geq \sup_Q \inf_W \mathbb{E}_{\ell_{1:T} \sim Q} [\text{Reg}(W, \ell_{1:T})] .$$

If we define $V_Q := \inf_W \mathbb{E}_{\ell_{1:T} \sim Q} [\text{Reg}(W, \ell_{1:T})]$ we can succinctly write, $V(\mathcal{W}, \mathcal{F}) = \sup_Q V_Q$.

Now, let us fix a distribution Q and denote the conditional expectation w.r.t. $\ell_{1:t}$ by $\mathbb{E}_t[\cdot]$ and the full expectation w.r.t. $\ell_{1:T}$ by $\mathbb{E}[\cdot]$. Substituting the definition of regret and noting that the infimum in its definition does not depend on the strategy W , we get

$$V_Q \geq \inf_W \left(\mathbb{E} \left[\sum_{t=1}^T \ell_t(W_t(\ell_{1:t-1})) \right] \right) - \mathbb{E} \left[\inf_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^T \ell_t(\mathbf{w}) \right] . \quad (6)$$

Let us simplify the infimum over \mathcal{P} -strategies as follows,

$$\begin{aligned} \inf_W \mathbb{E} \left[\sum_{t=1}^T \ell_t(W_t(\ell_{1:t-1})) \right] &= \inf_W \sum_{t=1}^T \mathbb{E} [\mathbb{E}_{t-1} [\ell_t(W_t(\ell_{1:t-1}))]] = \sum_{t=1}^T \inf_{W_t} \mathbb{E} [\mathbb{E}_{t-1} [\ell_t(W_t(\ell_{1:t-1}))]] \\ &= \sum_{t=1}^T \mathbb{E} \left[\inf_{\mathbf{w}_t \in \mathcal{W}} \mathbb{E}_{t-1} [\ell_t(\mathbf{w}_t)] \right] . \end{aligned}$$

Substituting this into (6), we get,

$$\begin{aligned} V_Q &\geq \sum_{t=1}^T \mathbb{E} \left[\inf_{\mathbf{w}_t \in \mathcal{W}} \mathbb{E}_{t-1} [\ell_t(\mathbf{w}_t)] \right] - \mathbb{E} \left[\inf_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^T \ell_t(\mathbf{w}) \right] \\ &= \mathbb{E} \left[\sup_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^T -\ell_t(\mathbf{w}) \right] - \sum_{t=1}^T \mathbb{E} \left[\sup_{\mathbf{w}_t \in \mathcal{W}} \mathbb{E}_{t-1} [-\ell_t(\mathbf{w}_t)] \right] . \end{aligned} \quad (7)$$

Since the losses ℓ_t are linear and \mathcal{W} is the unit ball, we can re-write the above as

$$V_Q \geq \mathbb{E} \left[\left\| \sum_{t=1}^T -\ell_t(\mathbf{w}) \right\| \right] - \sum_{t=1}^T \mathbb{E} [\|\mathbb{E}_{t-1} [-\ell_t(\mathbf{w}_t)]\|]$$

If we restrict ourselves to only distribution Q such that (ℓ_1, \dots, ℓ_T) are martingale difference sequences then clearly $\mathbb{E}_{t-1} [-\ell_t(\mathbf{w}_t)] = 0$ and so

$$\sup_Q V_Q \geq \sup_{\mathbf{M}} \mathbb{E} \left[\left\| \sum_{t=1}^T -\ell_t(\mathbf{w}) \right\| \right]$$

and so we get the lower bound. \square

Given the above result, we can now characterize the minimax value $V(U(\mathfrak{B}), \text{lin})$ in terms of the $p^*(\mathfrak{B}^*)$ where \mathfrak{B}^* is the dual space of \mathfrak{B} .

Theorem 4. *For all p, p' such that $p < p^*(\mathfrak{B}^*) < p'$, there exists a constant C such that,*

$$\Omega\left(T^{1/p'}\right) = V(U(\mathfrak{B}), \text{lin}) = V(U(\mathfrak{B}), \text{cvx}) \leq CT^{1/p}. \quad (8)$$

If the supremum in (3) is achieved, the upper bound also holds for $p = p^(\mathfrak{B}^*)$.*

Proof. To prove the lower bound, note that for any finite dimensional Banach space has $p^*(\mathfrak{B}^*) = 2$ with a possibly dimension dependent constant. In this case, the lower bound of \sqrt{T} for the linear game is easy: pick some non-zero vector, say $\ell \in U(\mathfrak{B}^*)$, and use ℓ or $-\ell$ at random in Theorem 3 and the lower bound follows. Therefore, assume \mathfrak{B} is infinite dimensional. The lower bound in this case is proved using Lemma 12 which in turn is proved using ideas from [16]. Lemma 12 shows that for any $p' > p^*(\mathfrak{B}^*)$,

$$\frac{\sup_{\mathbf{M}} \mathbb{E} \left[\left\| \sum_{t=1}^T \mathbf{d}_t \right\| \right]}{T^{1/p'}} \rightarrow \infty$$

However we have from Theorem 3 that

$$V(U(\mathfrak{B}), \text{lin}) \geq \sup_{\mathbf{M}} \mathbb{E} \left[\left\| \sum_{t=1}^T \mathbf{d}_t \right\| \right]$$

Hence we can conclude that for any $p' > p^*$ asymptotically $T^{1/p'}$ is dominated by $V(U(\mathfrak{B}), \text{lin})$ and hence the lower bound.

As for the upper bounds, if $p^*(\mathfrak{B}^*) = 1$ then the upper bound is trivial. On the other hand, when M-type is non-trivial, then M-type p implies M-cotype $q = \frac{p}{p-1}$. Therefore, for each $p \in (1, p^*(\mathfrak{B}^*))$, \mathfrak{B} is of M-cotype q . By Theorem (Pisier), there exists a q -uniformly convex function on \mathfrak{B} . Using this function in the Mirror Descent algorithm, Proposition 9 yields the required upper bound. \square

Although we have stated the above theorem for the special case when $\mathcal{W} = U(\mathfrak{B})$ and $\mathcal{F} = \text{lin}(U(\mathfrak{B}))$, it actually gives us regret rates when \mathcal{P} plays in $rU(\mathfrak{B})$ and \mathcal{A} plays L -Lipschitz linear functions via the following equality which is easy to prove from first principles,

$$V(rU(\mathfrak{B}), L \text{lin}(U(\mathfrak{B}))) = r \cdot L \cdot V(U(\mathfrak{B}), \text{lin}(U(\mathfrak{B}))) . \quad (9)$$

For the special case when $(\mathfrak{B}, \mathfrak{B}^*) = (\ell_q, \ell_p)$ for $1/p + 1/q = 1$, the rates given by the theorem above become $\Theta(T^{1/p})$ and $\Theta(T^{1/2})$ when $p \in (1, 2]$ and $p \in [2, \infty)$ respectively.

4 Convex-Bounded Games

Another natural game we consider is one in which \mathcal{P} plays from some convex set \mathcal{W} and \mathcal{A} plays some convex function *bounded* by 1. In the following theorem, we bound the value of such a game.

Theorem 5. *For all p, p' such that $p < p^*(\mathfrak{B}^*) < p'$, there is a constant C such that,*

$$\Omega\left(T^{1/p'}\right) \leq V(U(\mathfrak{B}), \text{bdd}) \leq CT^{1/p+1/2q} . \quad (10)$$

where $q = \frac{p}{p-1}$. If the supremum in (3) is achieved, the upper bound also holds for $p = p^(\mathfrak{B}^*)$.*

Proof of Theorem 5. Let us actually consider the case $\mathcal{W} = rU(\mathfrak{B})$ and $\mathcal{F} = \text{bdd}(rU(\mathfrak{B}))$. The bounds will turn out to be independent of r . Note that we have the inclusion, $\text{bdd}(rU(\mathfrak{B})) \supseteq \frac{1}{r} \text{lin}(U(\mathfrak{B}))$ which implies

$$V(rU(\mathfrak{B}), \text{bdd}(rU(\mathfrak{B}))) \geq V\left(rU(\mathfrak{B}), \frac{1}{r} \text{lin}(U(\mathfrak{B}))\right) .$$

The lower bound is now immediate due to lower bound on linear game on unit ball (Theorem 4) and property (9).

For the upper bound, note that any convex function bounded by 1 on the scaled ball $rU(\mathfrak{B})$ is $\frac{2}{\epsilon r}$ Lipschitz on the ball of radius $(1 - \epsilon)r$ [18]. Hence, by upper bound in Theorem 4 and property

(9), we see that there exists a strategy say W whose regret on the ball of radius $r(1 - \epsilon)$ is bounded by $\frac{C}{\epsilon} T^{\frac{1}{p}}$ for any $p \in [1, p^*(\mathfrak{B}^*)]$. That is

$$\sum_{t=1}^T \ell_t(W_t) - \operatorname{argmin}_{\mathbf{w} \in r(1-\epsilon)U(\mathfrak{B})} \sum_{t=1}^T \ell_t(\mathbf{w}) \leq \frac{C}{\epsilon} T^{1/p}, \quad \forall p \in [1, p^*(\mathfrak{B}^*)] \quad (11)$$

Let $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w} \in rU(\mathfrak{B})} \sum_{t=1}^T \ell_t(\mathbf{w})$. Now we consider two cases, first when $\mathbf{w}^* \in (1 - \epsilon)rU(\mathfrak{B})$. In this case the regret of the strategy on the unit ball is bounded by $\frac{C}{\epsilon} T^{1/p}$ for all $p \in [1, p^*(\mathfrak{B}^*)]$. On the other hand if $\mathbf{w}^* \notin (1 - \epsilon)rU(\mathfrak{B})$, then define $\mathbf{w}_\epsilon^* = \frac{r(1-\epsilon)\mathbf{w}^*}{\|\mathbf{w}^*\|}$. Note that $\mathbf{w}_\epsilon^* \in r(1 - \epsilon)U(\mathfrak{B})$. In this case by convexity of $\sum_{t=1}^T \ell_t(\mathbf{w})$, we have that

$$\sum_{t=1}^T \ell_t(\mathbf{w}_\epsilon^*) \leq \frac{r(1-\epsilon)}{\|\mathbf{w}^*\|} \sum_{t=1}^T \ell_t(\mathbf{w}^*) + \left(1 - \frac{r(1-\epsilon)}{\|\mathbf{w}^*\|}\right) \sum_{t=1}^T \ell_t(0)$$

Hence, we have that

$$\sum_{t=1}^T \ell_t(\mathbf{w}_\epsilon^*) - \sum_{t=1}^T \ell_t(\mathbf{w}^*) \leq \left(\frac{r(1-\epsilon)}{\|\mathbf{w}^*\|} - 1\right) \sum_{t=1}^T \ell_t(\mathbf{w}^*) + \left(1 - \frac{r(1-\epsilon)}{\|\mathbf{w}^*\|}\right) T \leq 2 \left(1 - \frac{r(1-\epsilon)}{\|\mathbf{w}^*\|}\right) T$$

However, since $\|\mathbf{w}^*\| \leq r$ we see that $\sum_{t=1}^T \ell_t(\mathbf{w}_\epsilon^*) - \sum_{t=1}^T \ell_t(\mathbf{w}^*) \leq 2\epsilon T$. Combining with (11) we see that for any $p \in [1, p^*(\mathfrak{B}^*)]$, $\sum_{t=1}^T \ell_t(\mathbf{w}_t) - \sum_{t=1}^T \ell_t(\mathbf{w}^*) \leq \frac{C}{\epsilon} T^{1/p} + 2\epsilon T$. Choosing $\epsilon = \sqrt{\frac{C}{2T^{1/q}}}$ we get the required upper bound. \square

Although we have stated the above result for the unit ball, the proof given above shows that the bounds are independent of the radius of the ball in which the player is playing.

Theorems 4 and 5 imply the following interesting corollary.

Corollary 6. *The following statements are equivalent :*

1. $V(U(\mathfrak{B}), \text{bdd}) = o(T)$.
2. $V(U(\mathfrak{B}), \text{cvx}) = o(T)$.
3. \mathfrak{B}^* has non-trivial M -type
4. Both \mathfrak{B} and \mathfrak{B}^* are super-reflexive.

Proof of Corollary 6. The implications **3** \Rightarrow **1** and **3** \Rightarrow **2** follow from the upper bounds in Theorems 4 and 5. The reverse implications **1** \Rightarrow **3** and **2** \Rightarrow **3**, in turn, follow from the lower bounds in those theorems. The equivalence of **3** and **4** is due to deep results of Pisier [19]. \square

The convex-Lipschitz games (and q -uniformly convex-Lipschitz games considered below) depend, by definition, not only on the player's set \mathcal{W} but also on the norm $\|\cdot\|$ of the underlying Banach space \mathfrak{B} . This is because \mathcal{A} 's functions are required to be Lipschitz w.r.t. $\|\cdot\|$. However, note that the convex-bounded game can be defined only in terms of the player set \mathcal{W} . Hence, one would expect the value of the game to be characterized solely by properties of set \mathcal{W} . This is what the following corollary confirms.

Corollary 7. *Let \mathcal{W} be any symmetric bounded convex subset of a vector space \mathbf{V} . The value of the bounded convex game on \mathcal{W} is non-trivial (i.e. $o(T)$) iff the Banach space $(\mathbf{V}, \|\cdot\|_{\mathcal{W}})$ (where $\|\cdot\|_{\mathcal{W}}$ is defined as in (5)) is super-reflexive.*

5 Uniformly Convex-Lipschitz Games

For any Hilbert space \mathfrak{H} , it is known that $V(U(\mathfrak{H}), \text{cvx}_{2,L})$ is much smaller than $V(U(\mathfrak{H}), \text{lin})$, i.e. the game is much easier for \mathcal{P} if \mathcal{A} plays 2-uniformly convex (also called *strongly convex*) functions. In fact, it is known that $V(U(\mathfrak{H}), \text{cvx}_{2,L}) = \Theta(L^2 \log T)$ while $V(U(\mathfrak{H}), \text{lin}) = \Theta(\sqrt{T})$. This suggests that we should get a rate between $\log(T)$ and \sqrt{T} if \mathcal{A} plays q -uniformly convex functions in a Hilbert space \mathfrak{H} for some $q > 2$. As far as we know, there is no characterization of the achievable rates for these intermediate situations even for Hilbert spaces. Our next result provides upper and lower bounds for $V(U(\mathfrak{B}), \text{cvx}_{q,L})$ in a Banach space, when the exponent of \mathcal{A} 's uniform convexity lies in an intermediate range between its minimum possible value q^* and its maximum value ∞ . It is easy to see that the minimum possible value q^* is $p^*(\mathfrak{B}^*)/(p^*(\mathfrak{B}^*) - 1)$.

Theorem 8. Let $q^* = \frac{p^*(\mathfrak{B}^*)}{p^*(\mathfrak{B}^*)-1}$ and $q \in (q^*, \infty)$. Let $p = q/(q-1)$ be the dual exponent of q . Then, as long as $\text{cvx}_{q,L}$ is non-empty, there exists K that depends on L such that for all $p' > p^*(\mathfrak{B}^*)$,

$$\Omega \left(\left(1 - \frac{1}{L}\right) \frac{1}{p} T^{1-p+\frac{p}{p'}} \right) \leq V(U(\mathfrak{B}), \text{cvx}_{q,L}) \leq KT^{\min\{2-p, 1/p^*(\mathfrak{B}^*)\}}. \quad (12)$$

Proof. We start by proving the lower bound. To this end note that if $\text{cvx}_{q,L}$ is non-empty, then the adversary plays L -Lipschitz, q -uniformly convex loss functions. Note that given such a function, there exists a norm $|\cdot|$ such that $|\cdot| \leq \|\cdot\| \leq L|\cdot|$ (ie. an equivalent norm) and $\frac{1}{q}|\cdot|^q$ is a q -uniformly convex function [20]. Given this we consider a game where adversary plays only functions from

$$\text{lincvx}_{q,L}(\mathcal{W}) := \left\{ \ell(\mathbf{w}) = \langle \mathbf{w}, \mathbf{x} \rangle + \frac{1}{q} |\mathbf{w}|^q : |x|_* \leq L - 1 \right\}$$

Note that since the above is L -Lipschitz w.r.t. $|\cdot|$, it is automatically L -Lipchitz w.r.t. $\|\cdot\|$. Hence $\text{lincvx}_{q,L} \subseteq \text{cvx}_{q,L}$, and so we have that $V(U(\mathfrak{B}), \text{lincvx}_{q,L}) \leq V(U(\mathfrak{B}), \text{cvx}_{q,L})$ However note that

$$V(U(\mathfrak{B}), \text{lincvx}_{q,L}) \geq \inf_W \sup_P \mathbb{E}_{\ell_{1:T} \sim P} \text{Reg}(W, \ell_{1:T}) \quad (13)$$

Also note that

$$\begin{aligned} \text{Reg}(W, \ell_{1:T}) &= \sum_{t=1}^T \left(\langle \mathbf{x}_t, \mathbf{w}_t \rangle + \frac{|\mathbf{w}_t|^q}{q} \right) - \inf_{\mathbf{w} \in U(\mathfrak{B})} \sum_{t=1}^T \left(\langle \mathbf{x}_t, \mathbf{w} \rangle + \frac{|\mathbf{w}|^q}{q} \right) \\ &= \sum_{t=1}^T \left(\langle \mathbf{x}_t, \mathbf{w}_t \rangle + \frac{|\mathbf{w}_t|^q}{q} \right) + T \sup_{\mathbf{w} \in U(\mathfrak{B})} \left(\left\langle -\frac{1}{T} \sum_{t=1}^T \mathbf{x}_t, \mathbf{w} \right\rangle - \frac{|\mathbf{w}|^q}{q} \right) \\ &\geq \sum_{t=1}^T \left(\langle \mathbf{x}_t, \mathbf{w}_t \rangle + \frac{\|\mathbf{w}_t\|^q}{Lq} \right) + T \sup_{\mathbf{w} \in U(\mathfrak{B})} \left(\left\langle -\frac{1}{T} \sum_{t=1}^T \mathbf{x}_t, \mathbf{w} \right\rangle - \frac{\|\mathbf{w}\|^q}{q} \right) \\ &= \sum_{t=1}^T \left(\langle \mathbf{x}_t, \mathbf{w}_t \rangle + \frac{\|\mathbf{w}_t\|^q}{Lq} \right) + \frac{T \left\| -\frac{1}{T} \sum_{t=1}^T \mathbf{x}_t \right\|_*^p}{p} \end{aligned}$$

Where the last step is by definition of convex dual of $\frac{\|\cdot\|^q}{q}$. Now note that since we have a supremum over distribution in (13), and so we can lower bound the value by picking distribution such that $\mathbf{d}_1, \dots, \mathbf{d}_T$ are martingale difference sequences and $\mathbf{d}_t \in \frac{L-1}{L} U(\mathfrak{B}^*)$. Thus we see that

$$\begin{aligned} V(U(\mathfrak{B}), \text{lincvx}_{q,L}) &\geq \sup_{\mathbf{M}} \left\{ \inf_W \mathbb{E} \left[\sum_{t=1}^T \left(\langle \mathbf{d}_t, \mathbf{w}_t \rangle + \frac{\|\mathbf{w}_t\|^q}{Lq} \right) + \frac{T^{1-p} \left\| -\sum_{t=1}^T \mathbf{d}_t \right\|_*^p}{p} \right] \right\} \\ &= \sup_{\mathbf{M}} \left\{ \inf_W \mathbb{E} \left[\sum_{t=1}^T \frac{\|\mathbf{w}_t\|^q}{Lq} \right] + \frac{T^{1-p} \mathbb{E} \left[\left\| -\sum_{t=1}^T \mathbf{d}_t \right\|_*^p \right]}{p} \right\} \\ &= \sup_{\mathbf{M}} \left\{ \frac{T^{1-p} \mathbb{E} \left[\left\| -\sum_{t=1}^T \mathbf{d}_t \right\|_*^p \right]}{p} \right\} \geq \frac{T^{1-p} \left(\sup_{\mathbf{M}} \mathbb{E} \left[\left\| -\sum_{t=1}^T \mathbf{d}_t \right\|_*^p \right] \right)^p}{p} \quad (14) \end{aligned}$$

where the first equality is because \mathbf{w}_t is only dependent on the history and so the conditional expectations over \mathbf{d}_t are 0 and the last step is due to Jensen's inequality. Now note that using Lemma 12 we see that for any $p' > p^*$ we have that

$$V(U(\mathfrak{B}), \text{lincvx}_{q,L}) = \Omega \left(\left(1 - \frac{1}{L}\right) \frac{T^{1-p+\frac{p}{p'}}}{p} \right) \quad (15)$$

(the $1 - 1/L$ term above comes from the fact that the martingale differences come from ball of radius $1 - 1/L$ while Lemma 12 is over unit ball). Note that the above lower bound becomes 0 when $L = 1$ but however in that case it means that the adversary is forced to play 1-Lipschitz, q -uniformly convex function. However since from each q -uniformly convex L -Lipschitz convex function we can

build an equivalent norm with distortion $1/L$, this means that the function the adversary plays can be used to construct the original norm itself. From the construction in [20] it becomes clear that the functions the adversary can play can be merely the norm plus some constant and so the lower bound of 0 is real.

Now we turn to proving the upper bound. Consider the regret of the mirror descent algorithm when we run it using a q^* -uniformly convex function Ψ that is C -Lipschitz on the unit ball. Here, for simplicity, we assume that the supremum is achieved in (3) (otherwise we can pick a Ψ that is q' -uniformly convex for $q' = p'/(p' - 1)$ where $p' = p^*(\mathfrak{B}^*) - 1/\log T$ and pay a constant factor). Note that in the case when $q > q^* + 1$, we have that each $\sigma_t^* = 0$ and so by Theorem 10,

$$\text{Reg}(\mathbf{w}_{1:T}, \ell_{1:T}) \leq 2 \min_{\lambda_1, \dots, \lambda_T} \sum_{t=1}^T \frac{(L+C)^{p^*}}{\left(\sum_{j \leq t} \lambda_j\right)^{p^*-1}} + 2 \sum_{t=1}^T \lambda_t C \leq 2 \min_{\lambda} \frac{(L+C)^{p^*} T^{2-p^*}}{\lambda^{p^*-1}} + 2T\lambda C$$

Using $\lambda = \frac{L+C}{T^{1/q^*}(2C)^{1/p^*}}$ we see that $\text{Reg}(\mathbf{w}_{1:T}, \ell_{1:T}) \leq 8(2C)^{1/q^*} (L+C) T^{1/p^*}$. On the other hand when $q \leq q^* + 1$, using the upper bound in the theorem with $\lambda_t = 0$ for all t we see that since all $q_t = q$ and all $\sigma_t^* = 1$ and $L_t = L$ we find that the regret of the adaptive algorithm is bounded as

$$\text{Reg}(\mathbf{w}_{1:T}, \ell_{1:T}) \leq \sum_{t=1}^T \left(\frac{2(L+C)^p}{t^{p-1}} + \frac{2(L+C)^{p^*}}{t^{p^*-1}} \right) \leq \sum_{t=1}^T \frac{4(L+C)^p}{t^{p-1}} \leq 4(L+C)^p \int_1^T \frac{1}{t^{p-1}} dt$$

Hence we see that for $p < p^*(\mathfrak{B}^*) \leq 2$, $\text{Reg}(\mathbf{w}_{1:T}, \ell_{1:T}) \leq \frac{4(L+C)^p}{2-p} T^{2-p}$. Since the regret of the adaptive algorithm bounds the value of the game, we see that by picking constant $K = \max\left\{\frac{4(L+C)^p}{2-p}, 8(2C)^{1/q^*} (L+C)\right\}$ we get the required upper bound. \square

The upper and lower bounds do not match in general and it is an interesting open problem to remove this gap. Note that the upper and lower bounds do match for the two extreme cases $q \rightarrow q^*$ and $q \rightarrow \infty$. When $q \rightarrow q^*$, then both lower and upper bound exponents tend to $2 - p^*(\mathfrak{B}^*)$. On the other hand, when $q \rightarrow \infty$, both exponents tend to $1/p^*(\mathfrak{B}^*)$.

6 Strategy for the Player

In this section, we provide a strategy known as Mirror Descent and is given as Algorithm 1 below which uses uniformly convex function Ψ as internal regularizer and is guaranteed to achieve low regret.

Algorithm 1 Mirror Descent (Parameters : $\eta > 0$, $\Psi : \mathfrak{B} \rightarrow \mathbb{R}$ which is uniformly convex)

for $t = 1$ to T **do**
 Play \mathbf{w}_t and receive ℓ_t
 $\mathbf{w}'_{t+1} \leftarrow \nabla \Psi^*(\nabla \Psi(\mathbf{w}_t) - \eta \boldsymbol{\lambda}_t)$ where $\boldsymbol{\lambda}_t \in \partial \ell_t(\mathbf{w}_t)$
 Update $\mathbf{w}_{t+1} \leftarrow \underset{\mathbf{w} \in \mathcal{W}}{\text{argmin}} \Delta_{\Psi}(\mathbf{w}, \mathbf{w}'_{t+1})$
end for

Example : As a more concrete example for the above strategy, if we consider the case where the player plays from the unit ball of a d dimensional ℓ_q space. In this case $\Psi(\mathbf{w}) = \frac{1}{q'} \|\mathbf{w}\|_q^{q'}$ where $q' = q$ whenever $q > 2$ and $q' = 2$ otherwise. The corresponding dual then is $\Psi^*(\mathbf{x}) = \frac{1}{p'} \|\mathbf{x}\|_p^{p'}$ where $p' = p$ if $q > 2$ and $p' = 2$ otherwise. Correspondingly we get

$$\begin{aligned} \nabla \Psi(\mathbf{w}) &= \|\mathbf{w}\|^{q'-q} (\text{sign}(\mathbf{w}_1)|\mathbf{w}_1|^{q-1}, \dots, \text{sign}(\mathbf{w}_d)|\mathbf{w}_d|^{q-1}) \quad \text{and} \\ \nabla \Psi^*(\mathbf{x}) &= \|\mathbf{x}\|^{q'-q} (\text{sign}(\mathbf{x}_1)|\mathbf{x}_1|^{q-1}, \dots, \text{sign}(\mathbf{x}_d)|\mathbf{x}_d|^{q-1}) \end{aligned}$$

We can use this to update \mathbf{w}'_{t+1} and get a concrete algorithm from the above strategy.

The following proposition gives a regret bound for Mirror Descent.

Proposition 9. *Suppose $\mathcal{W} \subseteq \mathfrak{B}$ is such that $\|\mathbf{w}\| \leq B$. Let MD denote the \mathcal{P} -strategy obtained by running Mirror Descent with a function Ψ that is q -uniformly convex on \mathfrak{B} and C -Lipschitz on \mathcal{W} , and the learning rate $\eta = (BC/T)^{1/p} \cdot (1/L)$. Here, $p = q/(q-1)$ is the dual exponent of q . Then, for all sequences $\ell_{1:T}$ such that ℓ_t is L -Lipschitz on \mathcal{W} , we have,*

$$\text{Reg}(\text{MD}, \ell_{1:T}) = O\left((BC)^{1/q} \cdot L \cdot T^{1/p}\right)$$

Proof. For $\lambda \in \mathfrak{B}^*$, $\mathbf{w} \in \mathfrak{B}$ we denote the pairing $\lambda(\mathbf{w})$ by $\langle \lambda, \mathbf{w} \rangle$ where $\langle \cdot, \cdot \rangle : \mathfrak{B}^* \times \mathfrak{B} \rightarrow \mathbb{R}$. This pairing is bilinear but not symmetric. We will first show that, for any $\mathbf{w} \in \mathcal{W}$,

$$\eta \langle \lambda_t, \mathbf{w}_t - \mathbf{w} \rangle \leq \Delta_\Psi(\mathbf{w}, \mathbf{w}_t) - \Delta_\Psi(\mathbf{w}, \mathbf{w}_{t+1}) + \frac{\eta^p}{p} \|\lambda_t\|_*^p, \quad (16)$$

where $p = q/(q-1)$. We have,

$$\begin{aligned} \eta \langle \lambda_t, \mathbf{w}_t - \mathbf{w} \rangle &= \langle \eta \lambda_t, \mathbf{w}_t - \mathbf{w}_{t+1} + \mathbf{w}_{t+1} - \mathbf{w} \rangle \\ &= \underbrace{\langle \eta \lambda_t, \mathbf{w}_t - \mathbf{w}_{t+1} \rangle}_{s_1} + \underbrace{\langle \eta \lambda_t + \nabla \Psi(\mathbf{w}_{t+1}) - \nabla \Psi(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w} \rangle}_{s_2} \\ &\quad + \underbrace{\langle \nabla \Psi(\mathbf{w}_t) - \nabla \Psi(\mathbf{w}_{t+1}), \mathbf{w}_{t+1} - \mathbf{w} \rangle}_{s_3} \end{aligned} \quad (17)$$

Now, by definition of the dual norm and the fact that $ab \leq \frac{a^p}{p} + \frac{b^q}{q}$ for any $a, b \geq 0$, we get

$$s_1 \leq \|(\mathbf{w}_t - \mathbf{w}_{t+1})\| \cdot \|\eta \lambda\|_* \leq \frac{1}{q} \|\mathbf{w}_t - \mathbf{w}_{t+1}\|^q + \frac{1}{p} \|\eta \lambda_t\|_*^p.$$

By the definition of the update, \mathbf{w}_{t+1} minimizes $\langle \eta \lambda_t - \nabla \Psi(\mathbf{w}_t), \mathbf{w} \rangle + \Psi(\mathbf{w})$ over $\mathbf{w} \in \mathcal{W}$. Therefore, $s_2 \leq 0$. Using simple algebraic manipulations, we get

$$s_3 = \Delta_\Psi(\mathbf{w}, \mathbf{w}_t) - \Delta_\Psi(\mathbf{w}, \mathbf{w}_{t+1}) - \Delta_\Psi(\mathbf{w}_{t+1}, \mathbf{w}_t).$$

Plugging this into (17), we get

$$\eta \langle \lambda_t, \mathbf{w}_t - \mathbf{w} \rangle \leq \Delta_\Psi(\mathbf{w}, \mathbf{w}_t) - \Delta_\Psi(\mathbf{w}, \mathbf{w}_{t+1}) + \frac{\eta^p}{p} \|\lambda_t\|_*^2 + \underbrace{\frac{1}{q} \|\mathbf{w}_t - \mathbf{w}_{t+1}\|^q - \Delta_\Psi(\mathbf{w}_{t+1}, \mathbf{w}_t)}_{s_4}$$

Using that Ψ is q -uniformly convex on \mathfrak{B} implies that $s_4 \leq 0$. So, we get (16).

We can now bound the regret as follows. For any $\mathbf{w} \in \mathcal{W}$, since $\lambda_t \in \partial \ell_t(\mathbf{w}_t)$, we have, $\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}) \leq \langle \lambda_t, \mathbf{w}_t - \mathbf{w} \rangle$. Combining this with (16) and summing over t gives,

$$\sum_{t=1}^T (\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w})) \leq \frac{\Delta_\Psi(\mathbf{w}, \mathbf{w}_1) - \Delta_\Psi(\mathbf{w}, \mathbf{w}_{T+1})}{\eta} + \frac{\eta^{p-1}}{p} \sum_{t=1}^T \|\lambda_t\|_*^p.$$

Now, $\Delta_\Psi(\mathbf{w}, \mathbf{w}_{T+1}) \geq 0$ and $\Delta_\Psi(\mathbf{w}, \mathbf{w}_1) \leq 2BC$. Further $\|\lambda_t\|_* \leq L$ since ℓ_t is L -Lipschitz. Plugging these above and optimizing over η gives the required upper bound. \square

Note that all the above algorithm needed to achieve low regret was a uniformly convex function Ψ . Theorem (Pisier) [16] gives us exactly this, it guarantees existence of a $\frac{p}{p-1}$ -uniformly convex function on a given Banach space for any $p \in [1, p^*(\mathfrak{B}^*))$ thus making sure that the above mirror descent algorithm with this choice of Ψ gives us the optimal rate for convex lipschitz games.

7 Adaptive Player Strategy

A natural extension of the q -uniformly convex lipschitz game is a game where at round t , \mathcal{A} plays q_t uniformly convex functions. In this section, we give an adaptive player strategy for such games that achieves the upper bound in Theorem 8 whenever the adversary plays only q -uniformly convex functions on all rounds and in general gets intermediate rates when the modulus of convexity on each round is different.

Now for the sake of readability, we assume that the supremum in (3) is achieved. The following theorem states that the same adaptive algorithm achieves the upper bound suggested in Theorem 8 for various q -uniformly convex games. Further the algorithm adjusts itself to the scenario when \mathcal{A} plays a different (σ_t, q_t) -uniformly convex function at each round t . To see this let, $\sigma_j^* = \sigma_j \mathbb{1}_{\{q_j < q^*+1\}}$. In the above algorithm we set at each round λ_t that satisfies,

$$2C\lambda_t = \sum_{i \leq t} \frac{\frac{\sigma_i^*}{M_i^{q_i}}}{\left(\sum_{j \leq t} \left[\frac{\sigma_j^*}{M_j^{q_j}} + \frac{\lambda_j}{M_t^{q^*}} \right] \right)^{p_i}} + \frac{1}{\left(\sum_{j \leq t} \left[\frac{\sigma_j^*}{M_j^{q_j}} + \frac{\lambda_j}{M_t^{q^*}} \right] \right)^{p^*-1}}, \quad (18)$$

where $M_t = L_t + C$. The following theorem which upper bounds the regret of the adaptive algorithm.

Algorithm 2 Adaptive Mirror Descent (Parameters : $\Psi : \mathfrak{B} \rightarrow \mathbb{R}$ which is q^* -uniformly convex)

$C \leftarrow$ Lipschitz constant of Ψ on $U(\mathfrak{B})$, $\mathbf{w}_1 \leftarrow \mathbf{0}$, $\Phi_1 \leftarrow \mathbf{0}$
for $t = 1$ to T **do**
 Play \mathbf{w}_t and receive ℓ_t which is L_t -Lipschitz and (σ_t, q_t) -uniformly convex
 Pick λ_t that satisfies (18)
 $\Phi_{t+1} \leftarrow \Phi_t + \ell_t + \lambda_t \Psi$
 $\mathbf{w}'_{t+1} \leftarrow \nabla \Phi_{t+1}^* (\nabla \Phi_t(\mathbf{w}_t))$
 Update $\mathbf{w}_{t+1} \leftarrow \operatorname{argmin}_{\mathbf{w} \in \mathcal{W}} \Delta_{\Phi_{t+1}}(\mathbf{w}, \mathbf{w}'_{t+1})$
end for

Theorem 10. Let $\mathcal{W} = U(\mathfrak{B})$. Let AMD denote the \mathcal{P} -strategy obtained by running Adaptive Mirror Descent with a Ψ which is $q^* = \frac{p^*(\mathfrak{B}^*)}{p^*(\mathfrak{B}^*)-1}$ uniformly convex. Then, for all sequences $\ell_{1:T}$ such that ℓ_t is L_t -Lipschitz and (σ_t, q_t) -uniformly convex, we have,

$$\operatorname{Reg}(\text{AMD}, \ell_{1:T}) \leq \min_{\lambda_{1:T}} \sum_{t=1}^T \left\{ \left(\sum_{i \leq t} \frac{\frac{2\sigma_i^*}{M_t^{q_i}}}{\left(\sum_{j \leq t} \left[\frac{\sigma_j^*}{M_t^{q_j}} + \frac{\lambda_j}{M_t^{q^*}} \right] \right)^{p_i}} \right) + \frac{2}{\left(\sum_{j \leq t} \left[\frac{\sigma_j^*}{M_t^{q_j}} + \frac{\lambda_j}{M_t^{q^*}} \right] \right)^{p^*-1}} + 2\lambda_t C \right\}$$

Proof. Note that $f_t = \ell_t + \lambda_t \Psi$, Ψ is q^* -uniformly convex and ℓ_t is (σ_t, q_t) -uniformly convex. Hence,

$$\Delta_{f_t}(\mathbf{w}'_{t+1}, \mathbf{w}_t) \geq \frac{\sigma_t^*}{q_t} \|\mathbf{w}'_{t+1} - \mathbf{w}_t\|^{q_t} + \frac{\lambda_t}{q^*} \|\mathbf{w}'_{t+1} - \mathbf{w}_t\|^{q^*}$$

Where $\sigma_t^* = \sigma_t \mathbb{I}_{\{q_t < q^*+1\}}$. Now since $\Phi_{t+1} = \sum_{i \leq t} f_t$, we see that

$$\begin{aligned} \Delta_{\Phi_{t+1}}(\mathbf{w}_t, \mathbf{w}'_{t+1}) &= \langle \nabla \Phi_{t+1}(\mathbf{w}_t) - \nabla \Phi_{t+1}(\mathbf{w}'_{t+1}), \mathbf{w}_t - \mathbf{w}'_{t+1} \rangle - \Delta_{\Phi_{t+1}}(\mathbf{w}'_{t+1}, \mathbf{w}_t) \\ &= \langle \nabla f_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}'_{t+1} \rangle - \Delta_{\Phi_{t+1}}(\mathbf{w}'_{t+1}, \mathbf{w}_t) \\ &\leq \langle \nabla f_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}'_{t+1} \rangle - \sum_{i=1}^t \frac{\sigma_i^*}{q_i} \|\mathbf{w}'_{i+1} - \mathbf{w}_i\|^{q_i} - \sum_{i=1}^t \frac{\lambda_i}{q^*} \|\mathbf{w}'_{i+1} - \mathbf{w}_i\|^{q^*} \end{aligned}$$

Now consider any arbitrary sequence $\beta_1, \dots, \beta_{2t}$ of non-negative numbers such that $\sum_{i=1}^{2t} \beta_i = 1$. In this case note that by Fenchel-Young inequality,

$$\begin{aligned} \Delta_{\Phi_{t+1}}(\mathbf{w}_t, \mathbf{w}'_{t+1}) &\leq \sum_{i=1}^{2t} \langle \beta_i \nabla f_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}'_{t+1} \rangle - \sum_{i=1}^t \frac{\sigma_i^*}{q_i} \|\mathbf{w}'_{i+1} - \mathbf{w}_i\|^{q_i} - \sum_{i=1}^t \frac{\lambda_i}{q^*} \|\mathbf{w}'_{i+1} - \mathbf{w}_i\|^{q^*} \\ &\leq \sum_{i=1}^t \left(\frac{\beta_i^{p_i} \|\nabla f_t(\mathbf{w}_t)\|_*^{p_i}}{p_i (\sigma_i^*)^{p_i/q_i}} + \frac{\beta_i^{p^*} \|\nabla f_t(\mathbf{w}_t)\|_*^{p^*}}{p^* \lambda_i^{p^*/q^*}} \right) \leq \sum_{i=1}^t \left(\frac{\beta_i^{p_i} (L_t + C)^{p_i}}{p_i (\sigma_i^*)^{p_i/q_i}} + \frac{\beta_i^{p^*} (L_t + C)^{p^*}}{p^* \lambda_i^{p^*/q^*}} \right) \end{aligned}$$

In the above we used the fact that since ℓ_t is L_t -Lipschitz and Ψ is C -Lipschitz so that, $\|\nabla f_t(\mathbf{w}_t)\|_* \leq (L_t + C)$ (we were able to get rid of the λ_t because we use $\lambda_t \leq 1$ and so $L_t + \lambda_t C \leq L_t + C$). Now choosing $\forall i \leq t$, $\beta_i \propto \frac{\sigma_i^*}{(L_t + C)^{q_i}}$ and $\forall t < i \leq 2t$, $\beta_i \propto \frac{\lambda_i}{(L_t + C)^{q^*}}$ we see that

$$\begin{aligned} \Delta_{\Phi_{t+1}}(\mathbf{w}_t, \mathbf{w}'_{t+1}) &\leq \sum_{i=1}^t \left(\frac{\frac{\sigma_i^*}{(L_t + C)^{q_i}}}{p_i \left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t + C)^{q_j}} + \frac{\lambda_j}{(L_t + C)^{q^*}} \right) \right)^{p_i}} + \frac{\frac{\lambda_i}{(L_t + C)^{q^*}}}{p^* \left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t + C)^{q_j}} + \frac{\lambda_j}{(L_t + C)^{q^*}} \right) \right)^{p^*}} \right) \\ &\leq \sum_{i=1}^t \left(\frac{\frac{\sigma_i^*}{(L_t + C)^{q_i}}}{\left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t + C)^{q_j}} + \frac{\lambda_j}{(L_t + C)^{q^*}} \right) \right)^{p_i}} + \frac{\frac{\lambda_i}{(L_t + C)^{q^*}} + \frac{\sigma_i^*}{(L_t + C)^{q_i}}}{\left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t + C)^{q_j}} + \frac{\lambda_j}{(L_t + C)^{q^*}} \right) \right)^{p^*}} \right) \\ &= \left(\sum_{i=1}^t \frac{\frac{\sigma_i^*}{(L_t + C)^{q_i}}}{\left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t + C)^{q_j}} + \frac{\lambda_j}{(L_t + C)^{q^*}} \right) \right)^{p_i}} \right) + \frac{1}{\left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t + C)^{q_j}} + \frac{\lambda_j}{(L_t + C)^{q^*}} \right) \right)^{p^*-1}} \end{aligned}$$

where in the first step we used the fact that $p^*, p_i \geq 1$ to remove them from the denominator. Thus using Lemma 13 we conclude that

$$\operatorname{Reg}(\mathbf{w}_{1:T}, \ell_{1:T}) \leq \sum_{t=1}^T \left(\sum_{i=1}^t \frac{\frac{\sigma_i^*}{(L_t + C)^{q_i}}}{\left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t + C)^{q_j}} + \frac{\lambda_j}{(L_t + C)^{q^*}} \right) \right)^{p_i}} + \frac{1}{\left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t + C)^{q_j}} + \frac{\lambda_j}{(L_t + C)^{q^*}} \right) \right)^{p^*-1}} + 2C\lambda_t \right)$$

Since we choose λ_t 's that satisfy Equation 18, using Lemma 14 we get the required statement. \square

Using the above regret bound we get the following corollary showing that the Adaptive Mirror Descent algorithm can be used to achieve all upper bounds on the regret presented in the paper.

Corollary 11. *There exists a Ψ which is q^* -uniformly convex function, and using this function with the Adaptive Mirror Descent (AMD) algorithm, we have the following.*

1. *Regret of AMD for convex-Lipschitz game matches upper bound in (8).*
2. *Regret of AMD for q -uniformly convex game matches upper bound in (12).*
3. *For the bounded convex game, there exists a $C > 0$ such that using AMD on $1 - CT^{-\frac{1}{2q^*}}$ ball achieves the upper bound in (10) for the game played on the unit ball.*

Proof. Claim 2 is shown in the constructive proof of the upper bound of Theorem 8. As for claim 1, note that this is the case of linear functions and so it is the same as adversary picking each $\sigma_t = 0$. Regret in this case again can be found in the proof of the upper bound of Theorem 8 and so claim 1 also holds. As for the last claim, given claim 1, it is evident from proof of Theorem 5. \square

When $q_1, \dots, q_T = 2$, AMD enjoys the same guarantee as Algorithm 4 in [7] (see Theorem 4.2).

8 Discussion

In future work, we also plan to convert the player strategies given here into implementable algorithms. Online learning algorithms can be implemented in infinite dimensional reproducing kernel Hilbert spaces [21] by exploiting the representer theorem and duality. We can, therefore, hope to implement online learning algorithms in infinite dimensional Banach spaces where some analogue of the representer theorem is available. Der and Lee [12] have made progress in this direction using the notion of *semi-inner products*. For $L_q(\Omega, \mu)$ spaces with q even, they showed how the problem of finding a maximum margin linear classifier can be reduced to a finite dimensional convex program using “moment functions”. The types (and their associated constants) of L_q spaces are well known from classical Banach space theory. So, we can use their ideas to get online algorithms in these spaces with provable regret guarantees. Vovk [4] also defines “Banach kernels” for certain Banach spaces of real valued functions and gives an implementable algorithm assuming the Banach kernel is efficiently computable. His interest is in prediction with the squared loss. It will be interesting to explore the connection of his ideas with the setting of this paper.

Using online-to-batch conversions, our results also imply error bounds for the estimation error in the batch setting. If $p^*(\mathfrak{B}^*) < 2$ then we get a rate worse than $O(T^{-1/2})$. However, we get the ability to work with richer function classes. This can decrease the approximation error. The study of this trade-off can be helpful.

We would also like to improve our lower and/or upper bounds where they do not match. In this regard, we should mention that the upper bound for the convex-bounded game given in Theorem 5 is not tight for a Hilbert space. Our upper bound is $O(T^{3/4})$ but it can be shown that using the self-concordant barrier $\log(1 - \|\mathbf{w}\|^2)$ for the unit ball, we get an upper bound of $O(T^{2/3})$.

Acknowledgments We thank the Colt 2010 reviewers for their helpful comments. We would also like to thank Maxim Raginsky for pointing out certain subtle issues in our earlier version.

References

- [1] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.
- [2] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2–3):169–192, 2007.
- [3] S. Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University, 2007.
- [4] V. Vovk. Competing with wild prediction rules. *Machine Learning*, 69(2–3):193–212, 2007.
- [5] J. Abernethy, P. L. Bartlett, A. Rakhlin, and A. Tewari. Optimal strategies and minimax lower bounds for online convex games. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2008.
- [6] A. Nemirovski and D. Yudin. *Problem complexity and method efficiency in optimization*. Nauka Publishers, Moscow, 1978.
- [7] P. L. Bartlett, E. Hazan, and A. Rakhlin. Adaptive online gradient descent. In *Advances in Neural Information Processing Systems 21*, 2007.

- [8] J. Abernethy, A. Agarwal, P. L. Bartlett, and A. Rakhlin. A stochastic view of optimal regret through minimax duality. In *Proceedings of the Twentieth Annual Conference on Computational Learning Theory*, 2009.
- [9] M. J. Donahue, L. Gurvits, C. Darken, and E. Sontag. Rates of convex approximation in non-Hilbert spaces. *Constructive Approximation*, 13:187–220, 1997.
- [10] L. Gurvits. A note on a scale-sensitive dimension of linear bounded functionals in Banach spaces. *Theoretical Computer Science*, 261(1):81–90, 2001.
- [11] S. Mendelson and G. Schechtman. The shattering dimension of sets of linear functionals. *The Annals of Probability*, 32(3):1746–1770, 2004.
- [12] R. Der and D. Lee. Large-margin classification in Banach spaces. In *Processing of Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.
- [13] Haizhang Zhang, Yuesheng Xu, and Jun Zhang. Reproducing kernel Banach spaces for machine learning. *Journal of Machine Learning Research*, 10:2741–2775, 2009.
- [14] Fedor Zhdanov, Alexey Chernov, and Yuri Kalnishkan. Aggregating algorithm competing with Banach lattices, 2010. arXiv preprint [arXiv:1002.0709](https://arxiv.org/abs/1002.0709) available at <http://arxiv.org/abs/1002.0709>.
- [15] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.
- [16] G. Pisier. Probabilistic methods in the geometry of Banach spaces. In G. Letta and M. Pratelli, editors, *Probability and Analysis*, volume 1206 of *Lecture Notes in Mathematics*, pages 167–241. Springer, 1986.
- [17] G. Pisier. Martingales with values in uniformly convex spaces. *Israel Journal of Mathematics*, 20(3–4):326–350, 1975.
- [18] C. Zălinescu. *Convex analysis in general vector spaces*. World Scientific Publishing Co. Inc., River Edge, NJ, 2002.
- [19] G. Pisier. Sur les espaces de Banach qui ne contiennent pas uniformément de ℓ_n^1 . *C. R. Acad. Sci. Paris Sér. A-B*, 277:A991–A994, 1973.
- [20] J. Borwein, A. J. Guirao, P. Hájek, and J. Vanderwerff. Uniformly convex functions on Banach spaces. *Proceedings of the American Mathematical Society*, 137(3):1081–1091, 2009.
- [21] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2002.
- [22] K. Sridharan and A. Tewari. Convex games in banach spaces. *Technical Report no. TTIC-TR-2009-6*, Toyota Technological Institute at Chicago, 2009.

Appendix

Lemma 12. For any Banach space \mathfrak{B}^* and any $p' > p^*(\mathfrak{B}^*)$ we have that

$$\frac{\sup_{\mathbf{M}} \mathbb{E} \left[\left\| \sum_{t=1}^T \mathbf{d}_t \right\| \right]}{T^{1/p'}} \rightarrow \infty$$

as $T \rightarrow \infty$, where \mathbf{M} refers to distributions over martingale difference sequences $(\mathbf{d}_t)_{t=1}^T$ such that each $\mathbf{d}_t \in U(\mathfrak{B}^*)$

Lemma 13. For the Adaptive Mirror Descent Algorithm we have that

$$\text{Reg}(\mathbf{w}_{1:T}, \ell_{1:T}) \leq \sum_{t=1}^T (\Delta_{\Phi_{t+1}}(\mathbf{w}_t, \mathbf{w}'_{t+1}) + 2\lambda_t C)$$

Lemma 14. Define for any sequence $\lambda_1, \dots, \lambda_S$ of any size S ,

$$O_S(\lambda_1, \dots, \lambda_S) = \sum_{t=1}^S \left(\sum_{i=1}^t \left(\frac{\frac{\sigma_i^*}{(L_t+C)^{q_i}}}{\left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t+C)^{q_j}} + \frac{\lambda_j}{(L_t+C)^{q^*}} \right) \right)^{p_i}} \right) + \frac{1}{\left(\sum_{j=1}^t \left(\frac{\sigma_j^*}{(L_t+C)^{q_j}} + \frac{\lambda_j}{(L_t+C)^{q^*}} \right) \right)^{p^*-1}} + 2C\lambda_t \right)$$

Then as long as we pick λ_t that satisfies Equation 18, we have that for any T

$$O_T(\lambda_1, \dots, \lambda_T) \leq 2 \min_{\lambda_1, \dots, \lambda_T} O\{\lambda_1, \dots, \lambda_T\}$$

For proofs of the above three lemma's refer to a longer version of this paper at [22].

Composite Objective Mirror Descent

John C. Duchi
UC Berkeley
jduchi@cs.berkeley.edu

Shai Shalev-Shwartz
Hebrew University
shais@cs.huji.ac.il

Yoram Singer
Google Research
singer@google.com

Ambuj Tewari
TTI Chicago
tewari@ttic.edu

Abstract

We present a new method for regularized convex optimization and analyze it under both online and stochastic optimization settings. In addition to unifying previously known first-order algorithms, such as the projected gradient method, mirror descent, and forward-backward splitting, our method yields new analysis and algorithms. We also derive specific instantiations of our method for commonly used regularization functions, such as ℓ_1 , mixed norm, and trace-norm.

1 Introduction and Problem Statement

Regularized loss minimization is a common learning paradigm in which one jointly minimizes an empirical loss over a training set plus a regularization term. The paradigm yields an optimization problem of the form

$$\min_{\mathbf{w} \in \Omega} \frac{1}{n} \sum_{t=1}^n f_t(\mathbf{w}) + r(\mathbf{w}), \quad (1)$$

where $\Omega \subset \mathbb{R}^d$ is the domain (a closed convex set), $f_t : \Omega \rightarrow \mathbb{R}$ is a (convex) loss function associated with a single example in a training set, and $r : \Omega \rightarrow \mathbb{R}$ is a (convex) regularization function. A few examples of famous learning problems that fall into this framework are least squares, ridge regression, support vector machines, support vector regression, lasso, and logistic regression.

In this paper, we describe and analyze a general framework for solving Eq. (1). The method we propose is a first-order approach, meaning that we access the functions f_t only by receiving subgradients. Recent work has shown that from the perspective of achieving good statistical performance on unseen data, first order methods are preferable to higher order approaches, especially when the number of training examples n is very large (Bottou and Bousquet, 2008; Shalev-Shwartz and Srebro, 2008). Furthermore, in large scale problems it is often prohibitively expensive to compute the gradient of the entire objective function (thus accessing all the examples in the training set), and randomly choosing a subset of the training set and computing the gradient over the subset (perhaps only a single example) can be significantly more efficient. This approach is very closely related to online learning. Our general framework handles both cases with ease—it applies to accessing a single example (or subset of the examples) at each iteration or accessing the entire training set at each iteration.

The method we describe is an adaptation of the Mirror Descent (MD) algorithm (Nemirovski and Yudin, 1983; Beck and Teboulle, 2003), an iterative method for minimizing a convex function $\phi : \Omega \rightarrow \mathbb{R}$. If the dimension d is large enough, MD is optimal among first-order methods, and it has a close connection to online learning since it is possible to bound the regret

$$\sum_{t=1}^T \phi_t(\mathbf{w}_t) - \inf_{\mathbf{w} \in \Omega} \sum_{t=1}^T \phi_t(\mathbf{w}),$$

where $\{\mathbf{w}_t\}$ is the sequence generated by mirror descent and the ϕ_t are convex functions. In fact, one can view popular online learning algorithms, such as weighted majority (Littlestone and Warmuth, 1994) and online gradient descent (Zinkevich, 2003) as special cases of mirror descent. A guarantee on the online regret can be translated directly to a guarantee on the convergence rate of the algorithm to the optimum of Eq. (1), as we will show later.

Following Beck and Teboulle’s exposition, a Mirror Descent update in the online setting can be written as

$$\mathbf{w}_{t+1} = \operatorname{argmin}_{\mathbf{w} \in \Omega} B_\psi(\mathbf{w}, \mathbf{w}_t) + \eta \langle \phi'_t(\mathbf{w}_t), \mathbf{w} - \mathbf{w}_t \rangle, \quad (2)$$

where B_ψ is a Bregman divergence and ϕ'_t denotes an arbitrary subgradient of ϕ_t . Intuitively, MD minimizes a first-order approximation of the function ϕ_t at the current iterate \mathbf{w}_t while forcing the next iterate \mathbf{w}_{t+1} to lie close to \mathbf{w}_t . The step-size η controls the trade-off between these two.

Our focus in this paper is to generalize mirror descent to the case when the functions ϕ_t are composite, that is, they consist of two parts: $\phi_t = f_t + r$. Here the f_t change over time but the function r remains constant. Of course, one can ignore the composite structure of the ϕ_t and use MD. However, doing so can result in undesirable effects. For example, when $r(\mathbf{w}) = \|\mathbf{w}\|_1$, applying MD directly does not lead to sparse updates. Since the sparsity inducing property of the ℓ_1 -norm is a major reason for its use. The modification of mirror descent that we propose is simple:

$$\mathbf{w}_{t+1} \triangleq \operatorname{argmin}_{\mathbf{w} \in \Omega} \eta \langle f'_t(\mathbf{w}_t), \mathbf{w} \rangle + B_\psi(\mathbf{w}, \mathbf{w}_t) + \eta r(\mathbf{w}). \quad (3)$$

This is almost the same as the mirror descent update with an important difference: we *do not* linearize r . We call this algorithm Composite Objective MIRROR Descent, or COMID. One of our contributions is to show that, in a variety of cases, the COMID update is no costlier than the usual mirror descent update. In these situations, each COMID update is efficient and benefits from the presence of the regularizer $r(\mathbf{w})$.

We now outline the remainder of the paper. We begin by reviewing related work, of which there is a copious amount, though we try to do some justice to prior research. We then give a general $O(\sqrt{T})$ regret bound for COMID in the online optimization setting, after which we give several extensions. We show $O(\log T)$ regret bounds for COMID when the composite functions $f_t + r$ are strongly convex, after which we show convergence rates and concentration results for stochastic optimization using COMID. The second focus of the paper is in the derived algorithms, where we outline step rules for several choices of Bregman function ψ and regularizer r , including ℓ_1 , ℓ_∞ , and mixed-norm regularization, as well as presenting new results on efficient matrix optimization with Schatten p -norms.

2 Related Work

Since the idea underlying COMID is simple, it is not surprising that similar algorithms have been proposed. One of our main contributions is to show that COMID generalizes much prior work and to give a clean unifying analysis. We do not have the space to thoroughly review the literature, though we try to do some small justice to what is known. We begin by reviewing work that we will show is a special case of COMID. Forward-backward splitting is a long-studied framework for minimizing composite objective functions (Lions and Mercier, 1979), though it has only recently been analyzed for the online and stochastic case (Duchi and Singer, 2009). Specializations of forward-backward splitting to the case where $r(\mathbf{w}) = \|\mathbf{w}\|_1$ include iterative shrinkage and thresholding from the signal processing literature (Daubechies et al., 2004), and from machine learning, Truncated Gradient (Langford et al., 2009) and SMIDAS (Shalev-Shwartz and Tewari, 2009) are both special cases of COMID.

In the optimization community there has been significant recent interest—both applied and theoretical—on minimization of composite objective functions such as that in Eq. (1). Some notable examples include Wright et al. (2009); Nesterov (2007); Tseng (2009). These papers all assume that the objective $f + r$ to be minimized is fixed and that f is smooth, i.e. that it has Lipschitz continuous derivatives. The most related of these to COMID is probably Tseng (2009, see his Sec. 3.1 and the references therein), which proposes the same update as ours, but gives a Nesterov-like optimal method for the fixed f case. We do not have restrictions on f , though by going to stochastic, nondifferentiable f we naturally suffer in convergence rate. Nonetheless, we do answer in the affirmative a question posed by Tseng (2009), which is whether stochastic or incremental subgradient methods work for composite objectives.

Two recent papers for online and stochastic composite objective minimization are Xiao (2009) and Duchi and Singer (2009). The former extends Nesterov’s 2009 analysis of primal-dual subgradient methods to the composite case, giving an algorithm which is similar to ours; however, our algorithms are different and the analysis for each is completely different. Duchi and Singer (2009) is simply a specialization of COMID to the case where the Euclidean Bregman divergence is used.

As a consequence of our general setting, we are able to give elegant new algorithms for minimization of functions on matrices, which include efficient and simple algorithms for trace-norm

minimization. Trace norm minimization has recently found strong applicability in matrix rank minimization (Recht et al., 2007), which has been shown to be very useful, for example, in collaborative filtering (Srebro et al., 2004). A special case of COMID has recently been developed for this task, which is very similar in spirit to fixed point and shrinkage methods from signal processing for ℓ_1 -minimization (Ma et al., 2009). The authors of this paper note that the method is extremely efficient for rank-minimization problems but do not give rates of convergence, which we give as a corollary to our main convergence theorems.

3 Notation and Setting

Before continuing, we establish notation and our problem setting formally. Vectors are lower case bold italic letters, such as $\mathbf{x} \in \mathbb{R}^d$, and scalars are lower case italics such as $x \in \mathbb{R}$. We denote a sequence of vectors by subscripts, i.e. $\mathbf{w}_t, \mathbf{w}_{t+1}, \dots$, and entries in a vector by non-bold subscripts as in w_j . Matrices are upper case bold italic letters, such as $\mathbf{W} \in \mathbb{R}^{d \times d}$. The subdifferential set of a function f evaluated at \mathbf{w} is denoted $\partial f(\mathbf{w})$ and a particular subgradient by $f'(\mathbf{w}) \in \partial f(\mathbf{w})$. When a function is differentiable, we write $\nabla f(\mathbf{w})$.

We focus mostly on the problem of regularized online learning, in which the goal is to achieve low regret w.r.t. a static predictor $\mathbf{w}^* \in \Omega$ on a sequence of functions $\phi_t(\mathbf{w}) \triangleq f_t(\mathbf{w}) + r(\mathbf{w})$. Here, f_t and $r \geq 0$ are convex functions, and Ω is some convex set (which could be \mathbb{R}^d). Formally, at every round of the algorithm we make a prediction $\mathbf{w}_t \in \mathbb{R}^d$ and then receive the function f_t . We seek bounds on the *regularized regret* with respect to \mathbf{w}^* , defined as

$$R_\phi(T, \mathbf{w}^*) \triangleq \sum_{t=1}^T [f_t(\mathbf{w}_t) + r(\mathbf{w}_t) - f_t(\mathbf{w}^*) - r(\mathbf{w}^*)]. \quad (4)$$

In batch optimization we set $f_t = f$ for all t , while in stochastic optimization we choose f_t to be the average of some random subset of $\{f_1, \dots, f_n\}$. As mentioned previously and as we will show, it is not difficult to transform regret bounds for Eq. (4) into convergence rates in expectation and with high probability for Eq. (1), which we do using techniques similar to Cesa-Bianchi et al. (2004).

Throughout, ψ designates a continuously differentiable function that is α -strongly convex w.r.t. a norm $\|\cdot\|$ on the set Ω . Recall that this means that the Bregman divergence associated with ψ ,

$$B_\psi(\mathbf{w}, \mathbf{v}) = \psi(\mathbf{w}) - \psi(\mathbf{v}) - \langle \nabla \psi(\mathbf{v}), \mathbf{w} - \mathbf{v} \rangle,$$

satisfies $B_\psi(\mathbf{w}, \mathbf{v}) \geq \frac{\alpha}{2} \|\mathbf{w} - \mathbf{v}\|^2$ for some $\alpha > 0$.

4 Composite Objective MIRROR Descent

We use proof techniques similar to those in Beck and Teboulle (2003) to derive “progress” bounds on each step of the algorithm. We then use the bounds to straightforwardly prove convergence results for online and batch learning. We begin by bounding the progress made by each step of the algorithm in either an online or a batch setting. This lemma is the key to our later analysis, so we prove it in full here.

Lemma 1 *Let the sequence $\{\mathbf{w}_t\}$ be defined by the update in Eq. (3). Assume that $B_\psi(\cdot, \cdot)$ is α -strongly convex with respect to a norm $\|\cdot\|$, that is, $B_\psi(\mathbf{w}, \mathbf{v}) \geq \frac{\alpha}{2} \|\mathbf{w} - \mathbf{v}\|^2$. For any $\mathbf{w}^* \in \Omega$,*

$$\eta(f_t(\mathbf{w}_t) - f_t(\mathbf{w}^*)) + \eta(r(\mathbf{w}_{t+1}) - r(\mathbf{w}^*)) \leq B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \frac{\eta^2}{2\alpha} \|f'_t(\mathbf{w}_t)\|_*^2.$$

Proof: The optimality of \mathbf{w}_{t+1} for Eq. (3) implies for all $\mathbf{w} \in \Omega$ and $r'(\mathbf{w}_{t+1}) \in \partial r(\mathbf{w}_{t+1})$,

$$\langle \mathbf{w} - \mathbf{w}_{t+1}, \eta f'_t(\mathbf{w}_t) + \nabla \psi(\mathbf{w}_{t+1}) - \nabla \psi(\mathbf{w}_t) + \eta r'(\mathbf{w}_{t+1}) \rangle \geq 0. \quad (5)$$

In particular, this obtains for $\mathbf{w} = \mathbf{w}^*$. From the subgradient inequality for convex functions, we have $f_t(\mathbf{w}^*) \geq f_t(\mathbf{w}_t) + \langle f'_t(\mathbf{w}_t), \mathbf{w}^* - \mathbf{w}_t \rangle$, or $f_t(\mathbf{w}_t) - f_t(\mathbf{w}^*) \leq \langle f'_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle$, and likewise for $r(\mathbf{w}_{t+1})$. We thus have

$$\begin{aligned} & \eta [f_t(\mathbf{w}_t) + r(\mathbf{w}_{t+1}) - f_t(\mathbf{w}^*) - r(\mathbf{w}^*)] \\ & \leq \eta \langle \mathbf{w}_t - \mathbf{w}^*, f'_t(\mathbf{w}_t) \rangle + \eta \langle \mathbf{w}_{t+1} - \mathbf{w}^*, r'(\mathbf{w}_{t+1}) \rangle \\ & = \eta \langle \mathbf{w}_{t+1} - \mathbf{w}^*, f'_t(\mathbf{w}_t) \rangle + \eta \langle \mathbf{w}_{t+1} - \mathbf{w}^*, r'(\mathbf{w}_{t+1}) \rangle + \eta \langle \mathbf{w}_t - \mathbf{w}_{t+1}, f'_t(\mathbf{w}_t) \rangle \\ & = \langle \mathbf{w}^* - \mathbf{w}_{t+1}, \nabla \psi(\mathbf{w}_t) - \nabla \psi(\mathbf{w}_{t+1}) - \eta f'_t(\mathbf{w}_t) - \eta r'(\mathbf{w}_{t+1}) \rangle + \langle \mathbf{w}^* - \mathbf{w}_{t+1}, \nabla \psi(\mathbf{w}_{t+1}) - \nabla \psi(\mathbf{w}_t) \rangle \\ & \quad + \eta \langle \mathbf{w}_t - \mathbf{w}_{t+1}, f'_t(\mathbf{w}_t) \rangle. \end{aligned}$$

Now, by Eq. (5), the first term in the last equation is non-positive. Thus we have that

$$\begin{aligned}
& \eta [f_t(\mathbf{w}_t) + r(\mathbf{w}_{t+1}) - f_t(\mathbf{w}^*) - r(\mathbf{w}^*)] \\
& \leq \langle \mathbf{w}^* - \mathbf{w}_{t+1}, \nabla \psi(\mathbf{w}_{t+1}) - \nabla \psi(\mathbf{w}_t) \rangle + \eta \langle \mathbf{w}_t - \mathbf{w}_{t+1}, f'_t(\mathbf{w}_t) \rangle \\
& = B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \eta \langle \mathbf{w}_t - \mathbf{w}_{t+1}, f'_t(\mathbf{w}_t) \rangle \\
& = B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \eta \left\langle \sqrt{\frac{\alpha}{\eta}}(\mathbf{w}_t - \mathbf{w}_{t+1}), \sqrt{\frac{\eta}{\alpha}} f'_t(\mathbf{w}_t) \right\rangle \\
& \leq B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \frac{\alpha}{2} \|\mathbf{w}_t - \mathbf{w}_{t+1}\|^2 + \frac{\eta^2}{2\alpha} \|f'_t(\mathbf{w}_t)\|_*^2 \\
& \leq B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \frac{\eta^2}{2\alpha} \|f'_t(\mathbf{w}_t)\|_*^2.
\end{aligned} \tag{6}$$

In the above, the first equality follows from simple algebra of B_ψ , that is, $\langle \nabla \psi(\mathbf{b}) - \nabla \psi(\mathbf{a}), \mathbf{c} - \mathbf{a} \rangle = B_\psi(\mathbf{c}, \mathbf{a}) + B_\psi(\mathbf{a}, \mathbf{b}) - B_\psi(\mathbf{c}, \mathbf{b})$ and setting $\mathbf{c} = \mathbf{w}^*$, $\mathbf{a} = \mathbf{w}_{t+1}$, and $\mathbf{b} = \mathbf{w}_t$. The second to last inequality follows from the Fenchel-Young inequality applied to the conjugate pair $\frac{1}{2} \|\cdot\|^2$, $\frac{1}{2} \|\cdot\|_*^2$ (Boyd and Vandenberghe, 2004, Example 3.27). The last inequality follows from the strong convexity of B_ψ with respect to the norm $\|\cdot\|$. \blacksquare

The following theorem uses Lemma 1 to establish a general regret bound for the COMID framework.

Theorem 2 *Let the sequence $\{\mathbf{w}_t\}$ be defined by the update in Eq. (3). Then for any $\mathbf{w}^* \in \Omega$,*

$$R_\phi(T, \mathbf{w}^*) \leq \frac{1}{\eta} B_\psi(\mathbf{w}^*, \mathbf{w}_1) + r(\mathbf{w}_1) + \frac{\eta}{2\alpha} \sum_{t=1}^T \|f'_t(\mathbf{w}_t)\|_*^2.$$

Proof: By Lemma 1,

$$\begin{aligned}
\eta \sum_{t=1}^T [f_t(\mathbf{w}_t) - f_t(\mathbf{w}^*) + r(\mathbf{w}_{t+1}) - r(\mathbf{w}^*)] & \leq \sum_{t=1}^T B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \frac{\eta^2}{2\alpha} \sum_{t=1}^T \|f'_t(\mathbf{w}_t)\|_*^2 \\
& = B_\psi(\mathbf{w}^*, \mathbf{w}_1) - B_\psi(\mathbf{w}^*, \mathbf{w}_{T+1}) + \frac{\eta^2}{2\alpha} \sum_{t=1}^T \|f'_t(\mathbf{w}_t)\|_*^2.
\end{aligned}$$

Noting that Bregman divergences are always non-negative, recall our assumption that $r(\mathbf{w}) \geq 0$. Adding $\eta r(\mathbf{w}_1)$ to both sides of the above equation and dropping the $r(\mathbf{w}_{t+1})$ term gives

$$\eta \sum_{t=1}^T [f_t(\mathbf{w}_t) - f_t(\mathbf{w}^*) + r(\mathbf{w}_t) - r(\mathbf{w}^*)] \leq B_\psi(\mathbf{w}^*, \mathbf{w}_1) + \eta r(\mathbf{w}_1) + \frac{\eta^2}{2\alpha} \sum_{t=1}^T \|f'_t(\mathbf{w}_t)\|_*^2.$$

Dividing each side by η gives the result. \blacksquare

A few corollaries are immediate from the above result. First, suppose that the functions f_t are Lipschitz continuous. Then there is some G_* such that $\|f'_t(\mathbf{w}_t)\|_* \leq G_*$. In this case, we have

Corollary 3 *Let $\{\mathbf{w}_t\}$ be generated by the update Eq. (3) and assume that the functions f_t are Lipschitz with dual Lipschitz constant G_* . Then*

$$R_\phi(T) \leq \frac{1}{\eta} B_\psi(\mathbf{w}^*, \mathbf{w}_1) + r(\mathbf{w}_1) + \frac{T\eta}{2\alpha} G_*^2.$$

If we take $\eta \propto 1/\sqrt{T}$, then we have a regret which is $O(\sqrt{T})$ when the functions f_t are Lipschitz. If Ω is compact, the f_t are guaranteed to be Lipschitz continuous (Rockafellar, 1970).

Corollary 4 *Suppose that either Ω is compact or the functions f_t are Lipschitz so $\|f'_t\|_* \leq G_*$. Also assume $r(\mathbf{w}_1) = 0$. Then setting $\eta = \sqrt{2\alpha B_\psi(\mathbf{w}^*, \mathbf{w}_1)}/(G_*\sqrt{T})$,*

$$R_\phi(T) \leq \sqrt{2TB_\psi(\mathbf{w}^*, \mathbf{w}_1)} G_* / \sqrt{\alpha}.$$

It is straightforward to prove results under the slightly different restriction that $\|f'_t(\mathbf{w})\|_*^2 \leq \rho f_t(\mathbf{w})$, which is similar to assuming a Lipschitz condition on the gradient of f_t . A common example in which this holds is linear regression, where $f_i(\mathbf{w}) = \frac{1}{2}(\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2$, so $\nabla f_i(\mathbf{w}) = (\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i) \mathbf{x}_i$ and $\rho = \frac{1}{2} \|\mathbf{x}_i\|^2$. The proof essentially amounts to dividing out constants dependent on η and ρ from both sides of the regret.

Corollary 5 Let $\|f'_t(\mathbf{w})\|_*^2 \leq \rho f_t(\mathbf{w})$, $r \geq 0$, and assume $r(\mathbf{w}_1) = 0$. Setting $\eta \propto 1/\sqrt{T}$ gives

$$R_\phi(T) = O(\rho\sqrt{T}B_\psi(\mathbf{w}^*, \mathbf{w}_1)/\alpha).$$

Proof: From Theorem 2, the non-negativity of r , and that $r(\mathbf{w}_1) = 0$ we immediately have

$$\sum_{t=1}^T \left(1 - \frac{\rho\eta}{2\alpha}\right) [f_t(\mathbf{w}_t) + r(\mathbf{w}_t)] \leq \sum_{t=1}^T \left(1 - \frac{\rho\eta}{2\alpha}\right) f_t(\mathbf{w}_t) + r(\mathbf{w}_t) \leq \frac{1}{\eta} B_\psi(\mathbf{w}^*, \mathbf{w}_1) + \sum_{t=1}^T f_t(\mathbf{w}^*) + r(\mathbf{w}^*)$$

Setting $\eta = 2\alpha/(\rho\sqrt{T})$ gives $1 - \rho\eta/(2\alpha) = (\sqrt{T} - 1)/\sqrt{T}$ so that

$$\sum_{t=1}^T f_t(\mathbf{w}_t) + r(\mathbf{w}_t) \leq \frac{\rho T}{2\alpha(\sqrt{T} - 1)} B_\psi(\mathbf{w}^*, \mathbf{w}_1) + \frac{\sqrt{T}}{\sqrt{T} - 1} \sum_{t=1}^T f_t(\mathbf{w}^*) + r(\mathbf{w}^*).$$

■

5 Logarithmic Regret for Strongly Convex Functions

Following the vein of research begun in Hazan et al. (2006) and Shalev-Shwartz and Singer (2007), we show that COMID can get stronger regret guarantees when we assume curvature of the loss functions f_t or r . Similar to Shalev-Shwartz and Singer, we now assume that for all t , $f_t + r$ is λ -strongly convex with respect to a differentiable function ψ , that is, for any $\mathbf{w}, \mathbf{v} \in \Omega$,

$$f_t(\mathbf{v}) + r(\mathbf{v}) \geq f_t(\mathbf{w}) + r(\mathbf{w}) + \langle f'_t(\mathbf{w}) + r'(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle + \lambda B_\psi(\mathbf{v}, \mathbf{w}). \quad (7)$$

For example, when $\psi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2$, we recover the usual definition of strong convexity. For simplicity, we assume that we push all the strong convexity into the function r so that the f_t are simply convex (clearly, this is possible by redefining $\hat{f}_t(\mathbf{w}) = f_t(\mathbf{w}) - \lambda\psi(\mathbf{w})$ if the f_t are λ -strongly convex). In this case, a straightforward corollary to Lemma 1 follows.

Corollary 6 Let the sequence $\{\mathbf{w}_t\}$ be defined by the update in Eq. (3) with step sizes η_t . Assume that $B_\psi(\cdot, \cdot)$ is α -strongly convex with respect to a norm $\|\cdot\|$ and that r is λ -strongly convex with respect to ψ . Then for any $\mathbf{w}^* \in \Omega$

$$\begin{aligned} & \eta_t (f_t(\mathbf{w}_t) - f_t(\mathbf{w}^*)) + \eta_t (r(\mathbf{w}_{t+1}) - r(\mathbf{w}^*)) \\ & \leq B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \frac{\eta_t^2}{2\alpha} \|f'_t(\mathbf{w}_t)\|_*^2 - \lambda\eta_t B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}). \end{aligned}$$

Proof: The proof is effectively identical to that of Lemma 1. We simply note that $r(\mathbf{w}_{t+1}) - r(\mathbf{w}^*) \leq \langle r'(\mathbf{w}_{t+1}), \mathbf{w}_{t+1} - \mathbf{w}^* \rangle - \lambda B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1})$ so that

$$\begin{aligned} & \eta_t [f_t(\mathbf{w}_t) + r(\mathbf{w}_{t+1}) - f_t(\mathbf{w}^*) - r(\mathbf{w}^*)] \\ & \leq \eta_t \langle \mathbf{w}_t - \mathbf{w}^*, f'_t(\mathbf{w}_t) \rangle + \eta_t \langle \mathbf{w}_{t+1} - \mathbf{w}^*, r'(\mathbf{w}_{t+1}) \rangle - \lambda\eta_t B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}). \end{aligned}$$

Now we simply proceed as in the proof of Lemma 1 following Eq. (5). ■

The above corollary almost immediately gives a logarithmic regret bound.

Theorem 7 Let r be λ -strongly convex with respect to a differentiable function ψ and suppose ψ is α -strongly convex with respect to a norm $\|\cdot\|$. Assume that $r(\mathbf{w}_1) = 0$. If $\|f'_t(\mathbf{w}_t)\|_* \leq G_*$ for all t ,

$$R_\phi(T) \leq \lambda B_\psi(\mathbf{w}^*, \mathbf{w}_1) + \frac{G_*^2}{\lambda\alpha} (\log T + 1) = O\left(\frac{G_*^2}{\lambda\alpha} \log T\right).$$

Proof: Rearranging Corollary 6, we have

$$\begin{aligned} & \sum_{t=1}^T f_t(\mathbf{w}_t) + r(\mathbf{w}_{t+1}) - f_t(\mathbf{w}^*) - r(\mathbf{w}^*) \\ & \leq \sum_{t=1}^T \left[\frac{1}{\eta_t} B_\psi(\mathbf{w}^*, \mathbf{w}_t) - \frac{1}{\eta_t} B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) - \lambda B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) \right] + \sum_{t=1}^T \frac{\eta_t}{2\alpha} \|f'_t(\mathbf{w}_t)\|_*^2 \\ & = \frac{1}{\eta_1} B_\psi(\mathbf{w}^*, \mathbf{w}_1) - \frac{1}{\eta_T} B_\psi(\mathbf{w}^*, \mathbf{w}_{T+1}) + \sum_{t=1}^{T-1} \left[B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t} \right) - \lambda B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) \right] \\ & \quad + \sum_{t=1}^T \frac{\eta_t}{2\alpha} \|f'_t(\mathbf{w}_t)\|_*^2 \end{aligned}$$

If we set $\eta_t = \frac{1}{\lambda t}$, then the first summation above is zero and

$$\sum_{t=1}^T f_t(\mathbf{w}_t) + r(\mathbf{w}_{t+1}) - f_t(\mathbf{w}^*) - r(\mathbf{w}^*) \leq \lambda B_\psi(\mathbf{w}^*, \mathbf{w}_1) + \frac{1}{\lambda\alpha} \sum_{t=1}^T \frac{1}{t} \|f'_t(\mathbf{w}_t)\|_*^2.$$

Noting that $\sum_{t=1}^T \frac{1}{t} \leq \log T + 1$ completes the proof of the theorem. \blacksquare

An interesting point regarding the above theorem is that we do not require $B_\psi(\mathbf{w}^*, \mathbf{w}_t)$ to be bounded or the set Ω to be compact, which previous work assumed. When the functions f_t are Lipschitz, then whenever r is strongly convex COMID still attains logarithmic regret.

Two notable examples attain the logarithmic bounds in the above theorem. It is clear that if r defines a valid Bregman divergence then that r is strongly convex with respect to itself in the sense of Eq. (7). First, consider optimization over the simplex with entropic regularization, that is, we set $r(\mathbf{w}) = \lambda \sum_i w_i \log w_i$ and $\Omega = \{\mathbf{w} : \mathbf{w} \succeq \mathbf{0}, \mathbf{1}^\top \mathbf{w} = 1\}$. In this case it is straightforward to see that $r(\mathbf{w}) = \sum_j w_j \log w_j$ is λ -strongly convex with respect to $\psi(\mathbf{w}) = r(\mathbf{w})$, which in turn is strongly convex with respect to the ℓ_1 -norm $\|\cdot\|_1$ over Ω (see Shalev-Shwartz and Singer, 2007, Definition 2 and Example 2). Since the dual of the ℓ_1 -norm is the ℓ_∞ norm, we have $R_\psi(T) = O\left(\frac{\log T}{\lambda} \max_t \|f'_t(\mathbf{w}_t)\|_\infty\right)$. We can also use $r(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2$, in which case we recover the same bounds as those in Hazan et al. (2006).

6 Stochastic Convergence Results

In this section, we examine the application of COMID to solving stochastic optimization problems. The techniques we use have a long history in online algorithms and make connections between the regret of the algorithm and generalization performance using martingale concentration results (Littlestone, 1989). We build on known techniques for data-driven generalization bounds (Cesa-Bianchi et al., 2004) to give concentration results for COMID in the stochastic optimization setting. Further work on this subject for the strongly convex case can be found in Kakade and Tewari (2008), though we focus on the case when $f_t + r$ is weakly convex.

We let $f(\mathbf{w}) = \mathbb{E}f(\mathbf{w}; Z) = \int f(\mathbf{w}; z) dP(z)$, and at every step t the algorithm receives an independent random variable $Z_t \sim P$ that gives an unbiased estimate $f_t(\mathbf{w}_t) = f(\mathbf{w}_t; Z_t)$ of the function f evaluated at \mathbf{w}_t and an unbiased estimate $f'_t(\mathbf{w}_t) = f'(\mathbf{w}_t; Z_t)$ of an arbitrary subgradient $f'(\mathbf{w}_t) \in \partial f(\mathbf{w}_t)$. We assume that $B_\psi(\mathbf{w}^*, \mathbf{w}_t) \leq D^2$ for all t and for simplicity that $\|f'_t(\mathbf{w}_t)\|_* \leq G_*$ for all t , which are satisfied when Ω is compact. We also assume without loss of generality that $r(\mathbf{w}_1) = 0$. For example, our original problem in which $f(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{w})$, where we randomly sample one f_i at each iteration, falls into this setup, resolving the question posed by Tseng (2009) on the existence of stochastic composite incremental subgradient methods.

Theorem 8 *Given the assumptions on f and Ω in the above paragraph, let $\{\mathbf{w}_t\}$ be the sequence generated by Eq. (3). In addition, let $\bar{\mathbf{w}}_T = \frac{1}{T} \sum_{t=1}^T \mathbf{w}_t$ and $\eta_t = \frac{D}{G_* \sqrt{\alpha t}}$. Then*

$$P\left(f(\bar{\mathbf{w}}_T) + r(\bar{\mathbf{w}}_T) \geq f(\mathbf{w}^*) + r(\mathbf{w}^*) + \frac{DG_*}{\sqrt{\alpha T}} + \varepsilon\right) \leq \exp\left(-\frac{T\alpha\varepsilon^2}{16D^2G_*^2}\right).$$

Alternatively, with probability at least $1 - \delta$

$$f(\bar{\mathbf{w}}_T) + r(\bar{\mathbf{w}}_T) \leq f(\mathbf{w}^*) + r(\mathbf{w}^*) + \frac{DG_*}{\sqrt{\alpha T}} \left(1 + 4\sqrt{\log \frac{1}{\delta}}\right).$$

Proof: We begin our derivation by recalling Lemma 1. Convexity of f and r imply

$$\begin{aligned} & \eta_t [f(\mathbf{w}_t) + r(\mathbf{w}_{t+1}) - f(\mathbf{w}^*) - r(\mathbf{w}^*)] \\ & \leq \eta_t \langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) \rangle + \eta_t \langle \mathbf{w}_{t+1} - \mathbf{w}^*, r'(\mathbf{w}_{t+1}) \rangle \\ & = \eta_t \langle \mathbf{w}_t - \mathbf{w}^*, f'_t(\mathbf{w}_t) \rangle + \eta_t \langle \mathbf{w}_{t+1} - \mathbf{w}^*, r'(\mathbf{w}_{t+1}) \rangle + \eta_t \langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) - f'_t(\mathbf{w}_t) \rangle \end{aligned}$$

We now follow the same derivation as Lemma 1, leaving $\eta_t \langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) - f'_t(\mathbf{w}_t) \rangle$ intact, thus

$$\begin{aligned} & \eta_t [f(\mathbf{w}_t) + r(\mathbf{w}_{t+1}) - f(\mathbf{w}^*) - r(\mathbf{w}^*)] \\ & \leq B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \frac{\eta_t^2}{2\alpha} \|f'_t(\mathbf{w}_t)\|_*^2 + \eta_t \langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) - f'_t(\mathbf{w}_t) \rangle. \end{aligned} \quad (8)$$

Now we subtract $r(\mathbf{w}_{T+1}) \geq 0$ from both sides, use the assumption that $r(\mathbf{w}_1) = 0$, and sum to get

$$\begin{aligned}
& \sum_{t=1}^T [f(\mathbf{w}_t) + r(\mathbf{w}_t) - f(\mathbf{w}^*) - r(\mathbf{w}^*)] \\
& \leq \sum_{t=1}^T \frac{1}{\eta_t} [B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1})] + \frac{1}{2\alpha} \sum_{t=1}^T \eta_t \|f'_t(\mathbf{w}_t)\|_*^2 + \sum_{t=1}^T \langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) - f'_t(\mathbf{w}_t) \rangle \\
& \leq \frac{1}{\eta_1} B_\psi(\mathbf{w}^*, \mathbf{w}_1) + \sum_{t=2}^T B_\psi(\mathbf{w}^*, \mathbf{w}_t) \left[\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right] + \frac{G_*^2}{2\alpha} \sum_{t=1}^T \eta_t + \sum_{t=1}^T \langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) - f'_t(\mathbf{w}_t) \rangle .
\end{aligned} \tag{9}$$

Let \mathcal{F}_t be a filtration with $Z_\tau \in \mathcal{F}_t$ for $\tau \leq t$. Since $\mathbf{w}_t \in \mathcal{F}_{t-1}$,

$$\mathbb{E}[\langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) - f'_t(\mathbf{w}_t; Z_t) \rangle \mid \mathcal{F}_{t-1}] = \langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) - \mathbb{E}[f'(\mathbf{w}_t; Z_t) \mid \mathcal{F}_{t-1}] \rangle = 0 ,$$

and thus the last sum in Eq. (9) is a martingale difference sequence. We next use our assumptions that $B_\psi(\mathbf{w}^*, \mathbf{w}_t) \leq D^2$ and $\frac{\alpha}{2} \|\mathbf{w}^* - \mathbf{w}_t\|^2 \leq B_\psi(\mathbf{w}^*, \mathbf{w}_t)$, therefore $\|\mathbf{w}^* - \mathbf{w}_t\| \leq \sqrt{2/\alpha}D$. Then

$$\langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) - f'_t(\mathbf{w}_t) \rangle \leq \|\mathbf{w}_t - \mathbf{w}^*\| \|f'(\mathbf{w}_t) - f'_t(\mathbf{w}_t)\|_* \leq 2\sqrt{2/\alpha}DG_* .$$

Thus Eq. (9) consists of a bounded difference martingale, and we can use standard concentration techniques to get strong convergence guarantees. Applying Azuma's inequality,

$$P\left(\sum_{t=1}^T \langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) - f'_t(\mathbf{w}_t) \rangle \geq \varepsilon\right) \leq \exp\left(-\frac{\alpha\varepsilon^2}{16TD^2G_*^2}\right) . \tag{10}$$

Define $\gamma_T = \sum_{t=1}^T \langle \mathbf{w}_t - \mathbf{w}^*, f'_t(\mathbf{w}_t) - f'(\mathbf{w}_t) \rangle$ and recall that $B_\psi(\mathbf{w}^*, \mathbf{w}_t) \leq D^2$. The convexity of f and r give $T[f(\bar{\mathbf{w}}_T) + r(\bar{\mathbf{w}}_T)] \leq \sum_{t=1}^T f(\mathbf{w}_t) + r(\mathbf{w}_t)$, so that

$$\begin{aligned}
T[f(\bar{\mathbf{w}}_T) + r(\bar{\mathbf{w}}_T)] & \leq T[f(\mathbf{w}^*) + r(\mathbf{w}^*)] + D^2 \left[\frac{1}{\eta_1} + \sum_{t=1}^T \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) \right] + \frac{G_*^2}{2\alpha} \sum_{t=1}^T \eta_t + \gamma_T \\
& = T[f(\mathbf{w}^*) + r(\mathbf{w}^*)] + \frac{D^2}{\eta_T} + \frac{G_*^2}{2\alpha} \sum_{t=1}^T \eta_t + \gamma_T .
\end{aligned}$$

Setting $\eta_t = \frac{D\sqrt{\alpha}}{G_*\sqrt{t}}$ we have $f(\bar{\mathbf{w}}_T) + r(\bar{\mathbf{w}}_T) \leq f(\mathbf{w}^*) + r(\mathbf{w}^*) + DG_*^2\sqrt{T}/\sqrt{\alpha} + \frac{1}{T}\gamma_T$, and we can immediately apply Azuma's inequality from Eq. (10) to complete the theorem. \blacksquare

7 Special Cases and Derived Algorithms

In this section, we show specific instantiations of our framework for different regularization functions r , and we also show that some previously developed algorithms are special cases of the framework for optimization presented here. We also give results on learning matrices with Schatten p -norm divergences that generalize some recent interesting work on trace norm regularization.

7.1 Fobos

The recently proposed FOBOS algorithm of Duchi and Singer (2009) is comprised, at each iteration, of the following two steps:

$$\tilde{\mathbf{w}}_{t+1} = \mathbf{w}_t - \eta f'_t(\mathbf{w}_t) \quad \text{and} \quad \mathbf{w}_{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w} - \tilde{\mathbf{w}}_{t+1}\|_2 + \eta r(\mathbf{w}) .$$

It is straightforward to verify that the update

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|_2^2 + \eta \langle f'_t(\mathbf{w}_t), \mathbf{w} - \mathbf{w}_t \rangle + \eta r(\mathbf{w})$$

is equivalent to the two step update above. Thus, COMID reduces to FOBOS when we take $\psi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2$ and $\Omega = \mathbb{R}^d$ (with constant learning rate η). This also shows that we can run FOBOS by restricting to a convex set $\Omega \neq \mathbb{R}^d$. Further, our results give tighter convergence guarantees than FOBOS, in particular, they do not depend in any negative way on the regularization function r .

It is also not difficult to show in general that the two step process of setting

$$\tilde{\mathbf{w}}_{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} B_\psi(\mathbf{w}, \mathbf{w}_t) + \eta \langle f'_t(\mathbf{w}_t), \mathbf{w} \rangle \quad \text{and} \quad \mathbf{w}_{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} B_\psi(\mathbf{w}, \tilde{\mathbf{w}}_t) + \eta r(\mathbf{w})$$

is equivalent to the original COMID update of Eq. (3) when $\Omega = \mathbb{R}^d$. Indeed, the optimal solution to the first step satisfies

$$\nabla\psi(\tilde{\mathbf{w}}_{t+1}) - \nabla\psi(\mathbf{w}_t) + \eta f'_t(\mathbf{w}_t) = 0 \quad \text{so that} \quad \tilde{\mathbf{w}}_{t+1} = \nabla\psi^{-1}(\nabla\psi(\mathbf{w}_t) - \eta f'_t(\mathbf{w}_t)).$$

Then looking at the optimal solution for the second step, for some $r'(\mathbf{w}_{t+1}) \in \partial r(\mathbf{w}_{t+1})$ we have

$$\nabla\psi(\mathbf{w}_{t+1}) - \nabla\psi(\tilde{\mathbf{w}}_{t+1}) + \eta r'(\mathbf{w}_{t+1}) = 0 \quad \text{i.e.} \quad \nabla\psi(\mathbf{w}_{t+1}) - \nabla\psi(\mathbf{w}_t) + \eta f'_t(\mathbf{w}_t) + \eta r'(\mathbf{w}_{t+1}) = 0.$$

This is clearly the solution to the one-step update of Eq. (3).

7.2 p -norm divergences

Now we consider divergence functions ψ which are the ℓ_p -norms squared. $\frac{1}{2} \|\mathbf{w}\|_p^2$ is $(p-1)$ -strongly convex over \mathbb{R}^d with respect to the ℓ_p -norm for any $p \in (1, 2]$ (Ball et al., 1994). We see that if we choose $\psi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_p^2$ to be the divergence function, we have a corollary to Theorem 2.

Corollary 9 *Suppose that $r(\mathbf{0}) = 0$ and that $\mathbf{w}_1 = \mathbf{0}$. Let $p = 1 + 1/\log d$ and use the Bregman function $\psi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_p^2$. Further suppose that either Ω is compact or the f_t are Lipschitz so that for $q = \log d + 1$, $\max_t \|f'_t(\mathbf{w}_t)\|_q \leq G_q$. Setting $\eta = \frac{\|\mathbf{w}^*\|_p}{G_q} \sqrt{\frac{1}{T \log d}}$, the regret of COMID satisfies*

$$R_\psi(T) \leq \|\mathbf{w}^*\|_p G_q \sqrt{T \log d} \asymp \|\mathbf{w}^*\|_1 G_\infty \sqrt{T \log d}.$$

Proof: Recall that the dual norm for an ℓ_p -norm is an ℓ_q -norm, where $q = p/(p-1)$. From Thm. 2, we immediately have that when $\mathbf{w}_1 = \mathbf{0}$ and $\psi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_p^2$

$$R(T) \leq \frac{1}{2\eta} \|\mathbf{w}^*\|_p^2 + \frac{\eta}{2(p-1)} \sum_{t=1}^T \|f'_t(\mathbf{w}_t)\|_q^2.$$

Now use the assumption that $\max_t \|f'_t(\mathbf{w}_t)\|_q \leq G_q$, replace p with $1 + 1/\log d$ (so $q = \log d + 1$), and set $\eta = c \sqrt{\frac{1}{T \log d}}$, which results in

$$R(T) \leq \frac{\sqrt{T \log d}}{2c} \|\mathbf{w}^*\|_p^2 + c \frac{\sqrt{T \log d}}{2} G_q^2.$$

Setting $c = \|\mathbf{w}^*\|_p / G_q$ gives us our desired result. \blacksquare

From the above, we see that COMID is a good candidate for (dense) problems in high dimensions, especially when we use ℓ_1 -regularization. For high dimensions when $\mathbf{w} \in \mathbb{R}^d$, taking $p = 1 + 1/\log d \approx 1$ means our bounds depend roughly on the ℓ_1 -norm of the optimal predictor and the infinity norm of the function gradients f_t . Shalev-Shwartz and Tewari (2009) recently proposed the ‘‘Stochastic Mirror Descent made Sparse’’ algorithm (SMIDAS) using this intuition. We recover SMIDAS by taking the divergence in COMID to be $\psi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_p^2$ and $r(\mathbf{w}) = \lambda \|\mathbf{w}\|_1$. The COMID update is

$$\nabla\psi(\tilde{\mathbf{w}}_{t+1}) = \nabla\psi(\mathbf{w}_t) - \eta f'_t(\mathbf{w}_t), \quad \mathbf{w}_{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} B_\psi(\mathbf{w}, \tilde{\mathbf{w}}_{t+1}) + \eta \lambda \|\mathbf{w}\|_1.$$

The SMIDAS update, on the other hand, is

$$\nabla\psi(\tilde{\mathbf{w}}_{t+1}) = \nabla\psi(\mathbf{w}) - \eta f'_t(\mathbf{w}_t), \quad \nabla\psi(\mathbf{w}_{t+1}) = \mathcal{S}_{\eta\lambda}(\nabla\psi(\tilde{\mathbf{w}}_{t+1})),$$

where \mathcal{S}_τ is the shrinkage/thresholding operator defined by

$$[\mathcal{S}_\tau(\mathbf{x})]_j = \operatorname{sign}(x_j) [|x_j| - \tau]_+ . \tag{11}$$

The following lemma proves that the two updates are identical in cases including p -norm divergences.

Lemma 10 *Suppose ψ is strongly convex and its gradient satisfies*

$$\operatorname{sign}([\nabla\psi(\mathbf{w})]_j) = \operatorname{sign}(w_j) . \tag{12}$$

Then the unique solution \mathbf{v} of $\mathbf{v} = \operatorname{argmin}_{\mathbf{w}} \{B_\psi(\mathbf{w}, \mathbf{u}) + \tau \|\mathbf{w}\|_1\}$ is given by

$$\nabla\psi(\mathbf{v}) = \mathcal{S}_\tau(\nabla\psi(\mathbf{u})) . \tag{13}$$

Proof: Since ψ is strongly convex, the solution is unique. We will show that if \mathbf{v} satisfies Eq. (13) then it is a solution to the problem. Therefore, suppose Eq. (13) holds. The proof proceeds by considering three cases.

Case I: $[\nabla\psi(\mathbf{u})]_j > \tau$. In this case, $[\nabla\psi(\mathbf{v})]_j = [\nabla\psi(\mathbf{u})]_j - \tau > 0$ and by Eq. (12), $v_j > 0$. Thus

$$[\nabla\psi(\mathbf{v})]_j - [\nabla\psi(\mathbf{u})]_j + \tau \text{sign}(v_j) = 0 .$$

Case II: $[\nabla\psi(\mathbf{u})]_j < -\tau$. In this case, $[\nabla\psi(\mathbf{v})]_j = [\nabla\psi(\mathbf{u})]_j + \tau < 0$ and Eq. (12) implies $v_j < 0$. So

$$[\nabla\psi(\mathbf{v})]_j - [\nabla\psi(\mathbf{u})]_j + \tau \text{sign}(v_j) = 0 .$$

Case III: $[\nabla\psi(\mathbf{u})]_j \in [-\tau, \tau]$. Here, we can take $v_j = 0$ and Eq. (12) will give $[\nabla\psi(\mathbf{v})]_j = 0$. Thus

$$0 \in [\nabla\psi(\mathbf{v})]_j - [\nabla\psi(\mathbf{u})]_j + \tau[-1, 1] .$$

Combining the three cases, \mathbf{v} satisfies $\mathbf{0} \in \nabla\psi(\mathbf{v}) - \nabla\psi(\mathbf{u}) + \tau\partial\|\mathbf{v}\|_1$, which is the optimality condition for $\mathbf{v} \in \text{argmin}_{\mathbf{w}}\{B_\psi(\mathbf{w}, \mathbf{u}) + \tau\|\mathbf{w}\|_1\}$. We thus have $\nabla\psi(\mathbf{v}) = \mathcal{S}_\tau(\nabla\psi(\mathbf{u}))$ as desired. ■

Rewriting the above lemma slightly gives the following result. The solution to

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}}{\text{argmin}} \{B_\psi(\mathbf{w}, \mathbf{w}_t) + \eta \langle f'_t(\mathbf{w}_t), \mathbf{w} - \mathbf{w}_t \rangle + \eta r(\mathbf{w})\}$$

when ψ satisfies the gradient condition of Eq. (12) is

$$\begin{aligned} \mathbf{w}_{t+1} &= (\nabla\psi)^{-1} [\text{sign}(\nabla\psi(\mathbf{w}_t) - \eta f'_t(\mathbf{w}_t)) \odot \max\{|\nabla\psi(\mathbf{w}_t) - \eta f'_t(\mathbf{w}_t)| - \eta\lambda, 0\}] \\ &= (\nabla\psi)^{-1} [\mathcal{S}_{\eta\lambda}(\nabla\psi(\mathbf{w}_t) - \eta f'_t(\mathbf{w}_t))] . \end{aligned} \quad (14)$$

Note that when $\psi(\cdot) = \|\cdot\|_p^2$ we recover Shalev-Shwartz and Tewari's SMIDAS, while with $p = 2$ we get Langford et al.'s 2009 truncated gradient method. See Shalev-Shwartz and Tewari (2009) or Gentile and Littlestone (1999) for the simple formulae to compute $(\nabla\psi)^{-1} \equiv \nabla\psi^*$.

ℓ_∞ -regularization Let us now consider the problem of setting $r(\mathbf{w})$ to be a general ℓ_p -norm (we will specialize this to ℓ_∞ shortly). We describe the dual function and then use it to derive a few particular updates, mentioning an open problem. First, let $p_1 > 1$ be the norm associated with the Bregman function ψ and p_2 be the norm for $r(\mathbf{w}) = \|\mathbf{w}\|_{p_2}$. Let $q_i = p_i/(p_i - 1)$ be the associated dual norm. Then, ignoring constants, the minimization problem from Eq. (3) becomes

$$\min_{\mathbf{w}} \langle \mathbf{v}, \mathbf{w} \rangle + \frac{1}{2} \|\mathbf{w}\|_{p_1}^2 + \lambda \|\mathbf{w}\|_{p_2} .$$

We introduce a variable $\mathbf{z} = \mathbf{w}$ and get the equivalent problem $\min_{\mathbf{w}=\mathbf{z}} \langle \mathbf{v}, \mathbf{w} \rangle + \frac{1}{2} \|\mathbf{w}\|_{p_1}^2 + \lambda \|\mathbf{z}\|_{p_2}$. To derive the dual of the problem, we introduce Lagrange multiplier $\boldsymbol{\theta}$ and find the Lagrangian

$$\mathcal{L}(\mathbf{w}, \mathbf{z}, \boldsymbol{\theta}) = \langle \mathbf{v} - \boldsymbol{\theta}, \mathbf{w} \rangle + \frac{1}{2} \|\mathbf{w}\|_{p_1}^2 + \lambda \|\mathbf{z}\|_{p_2} + \langle \boldsymbol{\theta}, \mathbf{z} \rangle .$$

Taking the infimum over \mathbf{w} and \mathbf{z} in the Lagrangian, since the conjugate of $\frac{1}{2} \|\cdot\|_p^2$ is $\frac{1}{2} \|\cdot\|_q^2$ when $1/p + 1/q = 1$ (Boyd and Vandenberghe, 2004, Example 3.27) we have

$$\inf_{\mathbf{w}} \left[\langle \mathbf{v} - \boldsymbol{\theta}, \mathbf{w} \rangle + \frac{1}{2} \|\mathbf{w}\|_{p_1}^2 \right] = -\frac{1}{2} \|\mathbf{v} - \boldsymbol{\theta}\|_{q_1}^2 \quad \inf_{\mathbf{z}} \left[\lambda \|\mathbf{z}\|_{p_2} + \langle \boldsymbol{\theta}, \mathbf{z} \rangle \right] = \begin{cases} 0 & \text{if } \|\boldsymbol{\theta}\|_{q_2} \leq \lambda \\ -\infty & \text{otherwise.} \end{cases}$$

Thus, our dual is the non-Euclidean projection problem

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \|\mathbf{v} - \boldsymbol{\theta}\|_{q_1} \quad \text{s.t.} \quad \|\boldsymbol{\theta}\|_{q_2} \leq \lambda .$$

The Lagrangian earlier is differentiable with respect to \mathbf{w} , so we can recover the optimal \mathbf{w} from $\boldsymbol{\theta}$ by noting that when $\psi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_{p_1}^2$, $\nabla\psi(\mathbf{w}) + \mathbf{v} - \boldsymbol{\theta} = 0$ at optimum or $\mathbf{w} = (\nabla\psi)^{-1}(\boldsymbol{\theta} - \mathbf{v})$. When $p_2 = 1$, we easily recover Eq. (14) as our update. However, the case $p_2 = \infty$ is more interesting, as it can be a building block for group-sparsity (Obozinski et al., 2007). In this case our problem is

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \|\mathbf{v} - \boldsymbol{\theta}\|_q \quad \text{s.t.} \quad \|\boldsymbol{\theta}\|_1 \leq \lambda .$$

It is clear by symmetry in the above that we can assume $\mathbf{v} \succeq \mathbf{0}$ with no loss of generality. We can raise the ℓ_q -norm to a power greater than 1 and maintain convexity, so our equivalent problem is

$$\min_{\boldsymbol{\theta}} \frac{1}{q} \|\mathbf{v} - \boldsymbol{\theta}\|_q^q \quad \text{s.t.} \quad \langle \mathbf{1}, \boldsymbol{\theta} \rangle \leq \lambda, \boldsymbol{\theta} \succeq \mathbf{0} . \quad (15)$$

Let $\widehat{\boldsymbol{\theta}}$ be the solution of Eq. (15). Clearly, at optimum we will have $\widehat{\theta}_i \leq v_i$, though we use this only for clarity in the derivation and omit constraints as they do not affect the optimization problem. Introducing Lagrange multipliers ν and $\boldsymbol{\alpha} \succeq \mathbf{0}$ we get the Lagrangian

$$\mathcal{L}(\boldsymbol{\theta}, \nu, \boldsymbol{\alpha}) = \frac{1}{q} \sum_{i=1}^d (v_i - \theta_i)^q + \nu(\langle \mathbf{1}, \boldsymbol{\theta} \rangle - \lambda) - \langle \boldsymbol{\alpha}, \boldsymbol{\theta} \rangle$$

Taking the derivative of the above, we have

$$-(v_i - \theta_i)^{q-1} + \nu - \alpha_i = 0 \quad \Rightarrow \quad \theta_i = v_i - (\nu - \alpha_i)^{1/(q-1)}$$

Now suppose we knew the optimal ν . If an index i satisfies $\nu \geq v_i^{q-1}$, then we will have $\widehat{\theta}_i = 0$. To see this, suppose for the sake of contradiction that $\widehat{\theta}_i > 0$. The KKT conditions for optimality of Eq. (15) (Boyd and Vandenberghe, 2004) imply that for such i we have

$$\widehat{\theta}_i = v_i - (\nu - \alpha_i)^{1/(q-1)} = v_i - \nu^{1/(q-1)} \leq 0,$$

a contradiction. Similarly, if $\nu < v_i^{q-1}$ and $\alpha_i \geq 0$, then $\widehat{\theta}_i > 0$. Ineed, since $\alpha_i \geq 0$,

$$\widehat{\theta}_i = v_i - (\nu - \alpha_i)^{1/(q-1)} > v_i - (v_i^{q-1} - \alpha_i)^{1/(q-1)} \geq v_i - v_i = 0,$$

so that $\widehat{\theta}_i > 0$ and the KKT conditions imply $\alpha_i = 0$. Had we known ν , the optimal $\widehat{\theta}_i$ would have been easy to attain as $\widehat{\theta}_i(\nu) = v_i - (\min\{\nu, v_i^{q-1}\})^{1/(q-1)}$ (note that this satisfies $0 \leq \widehat{\theta}_i \leq v_i$). Since we know that the structure of the optimal $\widehat{\boldsymbol{\theta}}$ must obey the above equation, we can boil our problem down to finding $\nu \geq 0$ so that

$$\sum_{i=1}^d \widehat{\theta}_i(\nu) = \sum_{i=1}^d v_i - \min\{\nu, v_i^{q-1}\}^{1/(q-1)} = \lambda. \quad (16)$$

Interestingly, this reduces to *exactly* the same root-finding problem as that for solving Euclidean projection to an ℓ_1 -ball (Duchi et al., 2008). As shown by Duchi et al., it is straightforward to find the optimal ν in time linear in the dimension d .

An open problem is to find an efficient algorithm for solving the generalized projections above when using the 2 rather than ∞ norm.

Mixed-norm regularization Now we consider the problem of mixed-norm regularization, in which we wish to minimize functions $f_t(\mathbf{W}) + r(\mathbf{W})$ of a matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$. In particular, we define $\bar{\mathbf{w}}_i \in \mathbb{R}^k$ to be the i^{th} row of \mathbf{W} , and we set $r(\mathbf{W}) = \lambda \sum_{i=1}^d \|\bar{\mathbf{w}}_i\|_{p_2}$. We also use the p -norm Bregman functions as above with $\psi(\mathbf{W}) = \frac{1}{2} \|\mathbf{W}\|_p^2 = \frac{1}{2} \left(\sum_{i,j} \mathbf{W}_{ij}^p \right)^{1/p}$, which are $(p-1)$ -strongly convex with respect to the ℓ_p -norm squared. As earlier, our minimization problem becomes

$$\min_{\mathbf{W}} \langle \mathbf{V}, \mathbf{W} \rangle + \frac{1}{2} \|\mathbf{W}\|_{p_1}^2 + \lambda \|\mathbf{W}\|_{\ell_1/\ell_{p_2}},$$

whose dual problem is

$$\min_{\boldsymbol{\Theta}} \|\mathbf{V} - \boldsymbol{\Theta}\|_{q_1} \quad \text{s.t.} \quad \|\bar{\boldsymbol{\theta}}_i\|_{q_2} \leq \lambda$$

Raising the first norm to the q_1 -power, we see that the problem is separable, and we can solve it using the techniques in the prequel.

7.3 Matrix Composite Mirror Descent

We now consider a setting that generalizes the previous discussions in which our variables \mathbf{W}_t are matrices $\mathbf{W}_t \in \Omega = \mathbb{R}^{d_1 \times d_2}$. We use Bregman functions based on Schatten p -norms (e.g. Horn and Johnson, 1985, Section 7.4). Schatten p -norms are the family of unitarily invariant matrix norms arising out of applying p -norms to the singular values of the matrix \mathbf{W} . That is, letting $\sigma(\mathbf{W})$ denote the vector of singular values of \mathbf{W} , we set $\|\mathbf{W}\|_p = \|\sigma(\mathbf{W})\|_p$. We use Bregman functions $\psi(\mathbf{W}) = \frac{1}{2} \|\mathbf{W}\|_p^2$, which, similar to the p -norm on vectors, are $(p-1)$ -strongly convex over $\Omega = \mathbb{R}^{d_1 \times d_2}$ with respect to the norm $\|\cdot\|_p$ (Ball et al., 1994).

As in the previous subsection, we mainly consider two values for p , $p = 2$ and a value very near 1, namely $p = 1 + 1/\log d$. For $p = 2$, ψ is 1-strongly convex with respect to $\|\cdot\|_2 = \|\cdot\|_{\text{F}}$, the

Frobenius norm. For the second value, ψ is $1/\log d$ -strongly convex with respect to $\|\cdot\|_{1+1/\log d}$, or, with a bit more work, ψ is $1/(3\log d)$ -strongly convex w.r.t. $\|\cdot\|_1$, the trace or nuclear norm.

We focus on the specific setting of trace-norm regularization, or $r(\mathbf{W}) = \lambda \|\mathbf{W}\|_1$. This norm, similar to the ℓ_1 -norm on vectors, gives sparsity in the singular value spectrum of \mathbf{W} and hence is useful for rank-minimization (Recht et al., 2007). The generic COMID update with the above choice of ψ gives a ‘‘Schatten p -norm’’ COMID algorithm for matrix applications:

$$\mathbf{W}_{t+1} = \underset{\mathbf{W} \in \Omega}{\operatorname{argmin}} \eta \langle f'_t(\mathbf{W}_t), \mathbf{W} \rangle + B_\psi(\mathbf{W}, \mathbf{W}_t) + \eta \lambda \|\mathbf{W}\|_1. \quad (17)$$

The update is well defined since ψ is strongly convex as per the above discussion. We also have defined $\langle \mathbf{W}, \mathbf{V} \rangle = \operatorname{tr}(\mathbf{W}^\top \mathbf{V})$ as the usual matrix inner product. The generic COMID convergence result in Thm. 2 immediately yields the following two corollaries.

Corollary 11 *Let the sequence $\{\mathbf{W}_t\}$ be defined by the update in Eq. (17) with $p = 2$. If each f_t satisfies $\|f'_t(\mathbf{W}_t)\|_2 \leq G_2$, then there is a stepsize η for which the regret against $\mathbf{W}^* \in \Omega$ is*

$$R_\phi(T) \leq G_2 \|\mathbf{W}^*\|_2 \sqrt{T}$$

Corollary 12 *Let $p = 1 + 1/\log d$ in the Schatten COMID update of Eq. (17). Let $q = 1 + \log d$. If each f_t satisfies $\|f'_t(\mathbf{W}_t)\|_q \leq G_q$ then*

$$R_\phi(T) \leq G_q \|\mathbf{W}^*\|_p \sqrt{T \log d} \asymp G_\infty \|\mathbf{W}^*\|_1 \sqrt{T \log d}$$

where $G_\infty = \max_t \|f'_t(\mathbf{W}_t)\|_\infty = \max_t \sigma_{\max}(f'_t(\mathbf{W}_t))$.

Let us consider the actual implementation of the COMID update. Similar convergence rates (with worse constants and a negative dependence on the spectrum of r) to those above can be achieved via simple mirror descent, i.e. by linearizing $r(\mathbf{W})$. The advantage of COMID is that it achieves sparsity in the spectrum, as the following proposition demonstrates.

Proposition 13 *Let $\Psi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_p^2$ and $\mathcal{S}_\tau(v) = \operatorname{sign}(v) [|v| - \tau]_+$ as in the prequel. For $p \in (1, 2]$, the update in Eq. (17) can be implemented as follows.*

$$\text{Compute SVD: } \mathbf{W}_t = \mathbf{U}_t \operatorname{diag}(\sigma(\mathbf{W}_t)) \mathbf{V}_t^\top \quad (18a)$$

$$\text{Gradient step: } \Theta_t = \mathbf{U}_t \operatorname{diag}(\nabla \Psi(\sigma(\mathbf{W}_t))) \mathbf{V}_t^\top - \eta f'_t(\mathbf{W}_t)$$

$$\text{Compute SVD: } \Theta_t = \tilde{\mathbf{U}}_t \operatorname{diag}(\sigma(\Theta_t)) \tilde{\mathbf{V}}_t^\top$$

$$\text{Splitting update: } \mathbf{W}_{t+1} = \tilde{\mathbf{U}}_t \operatorname{diag}((\nabla \Psi)^{-1}(\mathcal{S}_{\eta\lambda}(\sigma(\Theta_t)))) \tilde{\mathbf{V}}_t^\top \quad (18b)$$

Note that the first SVD, Eq. (18a), is used for notational convenience only and need not be computed at each iteration, since it is maintained at the end of iteration $t-1$ via Eq. (18b). The computational requirements for COMID are thus the same as standard mirror descent, which also requires an SVD computation on each step to compute $\partial \|\mathbf{W}\|_1$. The last step of the update, Eq. (18b) applies the shrinkage/thresholding operator $\mathcal{S}_{\eta\lambda}$ to the *spectrum* of Θ_t , which introduces sparsity. Furthermore, due to the sign (and hence sparsity) preserving nature of the map $(\nabla \Psi)^{-1}$, the sparsity in the spectrum is maintained in \mathbf{W}_{t+1} . Lastly, the special case for $p = 2$, the standard Frobenius norm update, was derived (but without rates or allowing stochastic gradients) by Ma et al. (2009), who report good empirical results for their algorithm. In trace-norm applications, we expect $\|\mathbf{W}^*\|_1$ to be small. Therefore, in such applications, our new Schatten- p COMID algorithm with $p \approx 1$ should give strong performance since G_∞ can be much smaller than G_2 .

Proof of Proposition 13: We know from the prequel that the COMID step is equivalent to

$$\nabla \psi(\tilde{\mathbf{W}}_t) = \nabla \psi(\mathbf{W}_t) - \eta f'_t(\mathbf{W}_t) \quad \text{and} \quad \mathbf{W}_{t+1} = \underset{\mathbf{W}}{\operatorname{argmin}} \left\{ B_\psi(\mathbf{W}, \tilde{\mathbf{W}}_t) + \eta r(\tilde{\mathbf{W}}_t) \right\}.$$

Since \mathbf{W}_t has singular value decomposition $\mathbf{U}_t \operatorname{diag}(\sigma(\mathbf{W}_t)) \mathbf{V}_t^\top$ and $\psi(\mathbf{W}) = \Psi(\sigma(\mathbf{W}))$ is unitarily invariant, $\nabla \psi(\mathbf{W}_t) = \mathbf{U}_t \operatorname{diag}(\nabla \Psi(\sigma(\mathbf{W}_t))) \mathbf{V}_t^\top$ (Lewis, 1995, Corollary 2.5). This means that the Θ_t computed in step 2 above is simply $\nabla \psi(\tilde{\mathbf{W}}_t)$. The proof essentially amounts to a reduction to the vector case, since the norms are unitarily invariant, and will be complete if we prove that

$$\mathbf{V} = \underset{\mathbf{W}}{\operatorname{argmin}} \left\{ B_\psi(\mathbf{W}, \tilde{\mathbf{W}}_t) + \tau \|\mathbf{W}\|_1 \right\}$$

has the unique solution

$$\mathbf{V} = \tilde{\mathbf{U}}_t \operatorname{diag} \left(\underbrace{(\nabla \Psi)^{-1}(\mathcal{S}_\tau(\sigma(\nabla \psi(\tilde{\mathbf{W}}_t)))}_{\tilde{\mathbf{w}}} \right) \tilde{\mathbf{V}}_t, \quad (19)$$

where $\tilde{\mathbf{U}}_t \operatorname{diag}(\sigma(\tilde{\mathbf{W}}_t)) \tilde{\mathbf{V}}_t^\top$ is the SVD of $\tilde{\mathbf{W}}_t$. By subgradient optimality conditions, it is sufficient that the proposed solution \mathbf{V} satisfy

$$\mathbf{0}_{d_1 \times d_2} \in \nabla \psi(\mathbf{V}) - \nabla \psi(\tilde{\mathbf{W}}_t) + \tau \partial \|\mathbf{V}\|_1.$$

Applying Lewis's Corollary 2.5, we can continue to use the orthonormal matrices $\tilde{\mathbf{U}}_t$ and $\tilde{\mathbf{V}}_t$, and we see that the proposed \mathbf{V} in Eq. (19) satisfies

$$\nabla \psi(\mathbf{V}) = \tilde{\mathbf{U}}_t \operatorname{diag}(\nabla \Psi(\tilde{\mathbf{w}})) \tilde{\mathbf{V}}_t^\top \quad \text{and} \quad \partial \|\mathbf{V}\|_1 = \tilde{\mathbf{U}}_t \operatorname{diag}(\partial \|\tilde{\mathbf{w}}\|_1) \tilde{\mathbf{V}}_t^\top.$$

We have thus reduced the problem to showing that $\mathbf{0}_d \in \nabla \Psi(\tilde{\mathbf{w}}) + \nabla \Psi(\sigma(\tilde{\mathbf{W}}_t)) + \tau \partial \|\tilde{\mathbf{w}}\|_1$, since we chose the matrices to have the same singular vectors by construction. From Lemma 10 presented earlier, we already know that $\tilde{\mathbf{w}}$ satisfies this equation if and only if $\nabla \Psi(\tilde{\mathbf{w}}) = \mathcal{S}_\tau(\nabla \Psi(\sigma(\tilde{\mathbf{W}}_t)))$, which is indeed the case by definition of $\tilde{\mathbf{w}}$ (noting of course that $\sigma(\nabla \psi(\tilde{\mathbf{W}}_t)) = \nabla \Psi(\sigma(\tilde{\mathbf{W}}_t))$). ■

Acknowledgments

JCD was supported by a National Defense Science and Engineering Graduate (NDSEG) fellowship, and some of this work was completed while JCD was an intern at Google Research.

References

- K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 68:357–367, 1967.
- K. Ball, E. Carlen, and E. Lieb. Sharp uniform convexity and smoothness inequalities for trace norms. *Inventiones mathematicae*, 115:463–482, 1994.
- A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 2003.
- L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems 20*, pages 161–168, 2008.
- S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, September 2004.
- I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communication on Pure and Applied Mathematics*, 57(11): 1413–1457, 2004.
- J. Duchi and Y. Singer. Online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.
- J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1 -ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.
- C. Gentile and N. Littlestone. The robustness of the p-norm algorithms. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.
- E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the 19th Annual Conference on Computational Learning Theory*, 2006.
- Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

- S. Kakade and A. Tewari. On the generalization ability of online strongly convex programming algorithms. In *Advances in Neural Information Processing Systems 22*. MIT Press, 2008.
- J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:747–776, 2009.
- A. Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 2:173–183, 1995.
- P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16:964–979, 1979.
- N. Littlestone. From on-line to batch learning. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 269–284, July 1989.
- Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.
- S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Submitted*, 2009. URL <http://arxiv.org/abs/0905.1643>.
- A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, New York, 1983.
- Y. Nesterov. Gradient methods for minimizing composite objective function. Technical Report 76, Center for Operations Research and Econometrics, Catholic University of Louvain (UCL), 2007.
- Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.
- G. Obozinski, B. Taskar, and M. Jordan. Joint covariate selection for grouped classification. Technical Report 743, Dept. of Statistics, University of California Berkeley, 2007.
- B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *Submitted*, 2007.
- R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.
- S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. Technical report, The Hebrew University, 2007. URL <http://www.cs.huji.ac.il/~shais>.
- S. Shalev-Shwartz and N. Srebro. SVM optimization: Inverse dependence on training set size. In *Proceedings of the 25th International Conference on Machine Learning*, pages 928–935, 2008.
- S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1 -regularized loss minimization. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.
- N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, 2004.
- P. Tseng. Approximation accuracy, gradient methods, and error bound for structured convex optimization. *Submitted*, 2009.
- S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.
- L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. In *Advances in Neural Information Processing Systems 23*, 2009.
- M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

Voting Paradoxes

Noga Alon, Tel Aviv University and Microsoft, Israel

Abstract

The early work of Condorcet in the 18th century, and that of Arrow and others in the 20th century, revealed the complex and interesting mathematical problems that arise in the theory of Social Choice, showing that the simple process of voting leads to strikingly counter-intuitive paradoxes. I will describe some of these, focusing on several recent intriguing examples whose analysis combines combinatorial and probabilistic ideas with techniques from the theory of the VC dimension of range spaces.

Optimal Algorithms for Online Convex Optimization with Multi-Point Bandit Feedback

Alekh Agarwal*
University of California
Berkeley, CA
alekh@cs.berkeley.edu

Ofer Dekel
Microsoft Research
Redmond, WA
oferd@microsoft.com

Lin Xiao
Microsoft Research
Redmond, WA
lin.xiao@microsoft.com

Abstract

Bandit convex optimization is a special case of online convex optimization with partial information. In this setting, a player attempts to minimize a sequence of adversarially generated convex loss functions, while only observing the value of each function at a single point. In some cases, the minimax regret of these problems is known to be strictly worse than the minimax regret in the corresponding full information setting. We introduce the multi-point bandit setting, in which the player can query each loss function at multiple points. When the player is allowed to query each function at two points, we prove regret bounds that closely resemble bounds for the full information case. This suggests that knowing the value of each loss function at two points is almost as useful as knowing the value of each function everywhere. When the player is allowed to query each function at $d + 1$ points (d being the dimension of the space), we prove regret bounds that are exactly equivalent to full information bounds for smooth functions.

1 Introduction

Online convex optimization is best understood as a repeated game between a player and an adversary. On round t of the game, the player begins by choosing a point x_t from a fixed and known convex set $\mathcal{K} \subseteq \mathbb{R}^d$. We adopt the game-theoretic terminology and say that x_t is the player's *move* on round t . The adversary observes x_t and chooses a convex loss function $\ell_t : \mathcal{K} \rightarrow \mathbb{R}$. Then, the loss function ℓ_t is revealed to the player, who incurs the loss $\ell_t(x_t)$. The goal of the player is to minimize his *regret*, defined as

$$\sum_{t=1}^T \ell_t(x_t) - \min_{x \in \mathcal{K}} \sum_{t=1}^T \ell_t(x) .$$

Regret measures the difference between the cumulative loss of the player's strategy and the loss of the best constant point chosen in hindsight. Although the adversary has the advantage of playing second on each round, there exist strategies for which we can prove non-trivial upper-bounds on regret. Zinkevich (2003) presents a strategy based on gradient descent that guarantees a regret of $O(\sqrt{T})$ when the set \mathcal{K} is compact and the loss functions are Lipschitz continuous. If the loss functions are strongly convex (defined below), Hazan et al. (2007) analyze a similar gradient descent strategy with a regret bound of $O(\log(T))$. They also prove a similar regret bound for the larger class of exp-concave loss functions using an online Newton-step algorithm. The $O(\sqrt{T})$ and $O(\log(T))$ regret bounds, for convex and strongly convex loss functions respectively, are known to be minimax optimal (see eg. Abernethy et al. (2009); Cesa-Bianchi and Lugosi (2006) and references therein).

The online convex optimization problem becomes more challenging when the player only receives partial feedback on the choices of the adversary. One particularly interesting type of feedback is *bandit* feedback, where the adversary only reveals the value of the loss function at x_t , instead of revealing the entire function ℓ_t . Specifically, the player does not know the gradient of ℓ_t at x_t and a simple gradient descent algorithm is inapplicable.

In the bandit setting, the player does not stand a chance against the *completely adaptive* adversary described above, who chooses ℓ_t after observing the player's move. Specifically, a regret bound for

*This research was conducted while AA was a research intern in Microsoft Research at Redmond. AA gratefully acknowledges the support of the NSF through grants 0707060 and 0830410 for travel expenses

Algorithm 1 Template for k -point bandit algorithm

for $t = 1, \dots, T$ **do**
 Adversary (secretly) chooses convex loss function ℓ_t .
 Player chooses and reveals $y_{t,1}, \dots, y_{t,k} \in \mathcal{K}$.
 Adversary reveals $\ell_t(y_{t,1}), \dots, \ell_t(y_{t,k})$.
 Player incurs the loss $(1/k) \sum_{i=1}^k \ell_t(y_{t,i})$.
end for

any strategy in this setting is necessarily $\Omega(T)$. To see this, let \mathcal{K} be the interval $[0, 1]$ and define the following two completely adaptive adversaries: the first chooses $\ell_t(z) = z - x_t$ while the second chooses $\ell'_t(z) = x_t - z$. In both cases, the player observes the loss value $\ell_t(x_t) = \ell'_t(x_t) = 0$ on every round, and has no way of knowing which of the two adversaries he is facing. Therefore, the player will play the same sequence of moves against both adversaries. If at least half of the player's moves lie in the interval $[\frac{1}{2}, 0]$ then the player's regret against the first adversary is at least $T/4$. Otherwise, the player's regret against the second adversary is at least $T/4$. Overall, any strategy will suffer at least a linear regret against one of these adversaries.

The example above indicates that we need to level the playing field by slightly limiting the power of the adversary. Therefore, an *adaptive* adversary in the bandit setting is allowed to choose ℓ_t based only on the player's past moves x_1, \dots, x_{t-1} , and not on his current move x_t . Put another way, the adversary chooses ℓ_t at the beginning of round t , before the player makes his move. Then, the player chooses x_t without knowing ℓ_t and reveals his move to the adversary. Finally, the adversary notifies the player of his loss $\ell_t(x_t)$. We note in passing that an even weaker adversary is the *oblivious* adversary, who chooses ℓ_t without knowing any of the player's moves. However, in practice, an analysis for the oblivious case is often an intermediate step towards an analysis for the adaptive case.

In the special case where the loss functions are linear, Abernethy and Rakhlin (2009) present a randomized algorithm with an $\tilde{O}(\sqrt{T})$ regret-bound that holds with high-probability against adaptive adversaries. This algorithm relies on a non-trivial application of self-concordant barrier regularization, and differs significantly from the typical algorithms that are used in the full information case. This $\tilde{O}(\sqrt{T})$ regret bound is optimal due to an information-theoretic argument from Auer et al. (2003). For general convex loss functions, Flaxman et al. (2005) present a simple modification of the full information gradient descent algorithm, where the gradient is replaced by a randomized estimate. The expected regret of this algorithm is shown to be $O(T^{3/4})$ against oblivious adversaries. Flaxman et al. also sketch a high probability extension of their bound to adaptive adversaries. For smooth and strongly convex loss functions, the regret bound of Flaxman et al. (2005) can be strengthened to $O(T^{2/3})$, and furthermore, if \mathcal{K} is a linear vector space (namely, the optimization is unconstrained) then the bound can be improved to $O(\sqrt{T})$ ¹. Finding an optimal algorithm for the general bandit convex optimization setting remains an open problem. However, a lower bound due to Dani et al. (2008) implies that the regret of this optimal algorithm will be $\Omega(\sqrt{T})$, even when the functions are strongly convex. This is to be contrasted with the full-information case where $O(\log(T))$ regret is achieved by online gradient descent when the loss functions are strongly convex.

Overall, we observe significant gaps between the optimal regret bounds for the full information and bandit settings, as well as gaps in the complexity of the algorithms that attain these bounds. This leads to the natural question of whether we can study a continuum of problems ranging between these two extremes. Through such an inquiry, we can hope to find the point along this continuum where the regret bounds for the full information case deteriorate to become the inferior bounds in the bandit setting. We can also hope to find the point on this continuum where simple gradient descent approaches stop giving optimal bounds, and the need for specialized algorithms arises. Answering these questions is the focus of our paper.

To this end, we extend the bandit setting and introduce the multi-point bandit setting, where the player queries each loss function at k randomized points, rather than at a single point. The template for a k -point bandit algorithm is given in Algorithm 1. In this setting, we define the expected regret,

$$\mathbb{E} \frac{1}{k} \sum_{t=1}^T \sum_{i=1}^k \ell_t(y_{t,i}) - \min_{x \in \mathcal{K}} \mathbb{E} \sum_{t=1}^T \ell_t(x),$$

where expectation is taken over the randomness of the player. When $k = 1$, the multi-point bandit

¹Both of these results are novel but their proof is omitted due to space constraints.

setting reduces to the bandit case discussed above.

For $k = 2$, we show that a variant of the randomized gradient descent algorithm proposed by Flaxman et al. (2005) performs optimally. We prove a *high-probability* regret bound of $\tilde{O}(\sqrt{T})$ for convex Lipschitz-continuous loss functions chosen by an adaptive adversary. We also prove an *expected* regret bound of $O(\log(T))$ for strongly convex loss functions chosen by an adaptive adversary. Thus, by allowing the player to query only two points on each round, we can already obtain bounds that closely resemble the optimal bounds for the full-information setting. In the strongly convex case, we see a sharp phase transition of the minimax regret from $\Omega(\sqrt{T})$ to $O(\log(T))$ when moving from $k = 1$ to $k = 2$. For general convex loss functions, optimal bounds for the $k = 1$ setting achievable in a computationally efficient manner are not yet known, but it seems evident that optimal algorithms and bounds will rely on techniques that are significantly different from those used in the full information-case. In contrast, for $k = 2$ we can rely on algorithms and analysis techniques that are quite similar to those used in the full information case. Overall, we conclude that having the ability to evaluate the loss function at two points on each round is *almost* as powerful as observing the entire loss function.

When $k = d + 1$, where d is the dimension of the space, we show that a *deterministic* algorithm can obtain a regret of $O(\sqrt{T})$ against convex Lipschitz and smooth loss functions, and a regret of $O(\log(T))$ against strongly convex and smooth loss functions. Moreover, both of these bounds hold for the case of *completely adaptive* adversaries, precisely as in the full information case. This algorithm uses gradient descent based on a deterministic approximation of the gradient. Applying a similar approximation to the online Newton-step algorithm (Hazan et al., 2007) also gives an $O(\log(T))$ regret for exp-concave and smooth loss functions. We conclude that having the ability to evaluate the loss function at $d + 1$ points on each round is as powerful as observing the entire function when the functions are smooth.

The algorithms we propose are efficient, with the computational complexities being no greater than their full information counterparts.

Similar schemes for constructing gradient estimators using multiple function evaluations have also been analyzed in the framework of stochastic optimization (Polyak & Tsypkin, 1973; Spall, 2003) and derivative-free optimization (Conn et al., 2009). However, there seem to be relatively few relevant results concerning the rate of convergence.

Notation and Assumptions: Before proceeding, we define our notation and state assumptions that will appear in various parts of the analysis. Let $\|\cdot\|$ denote the Euclidean norm in \mathbb{R}^d , and let $\mathcal{B} = \{x \in \mathbb{R}^d : \|x\| \leq 1\}$ be the unit ball centered at the origin. We assume that the convex set \mathcal{K} is compact and has a nonempty interior. If not, we can always map \mathcal{K} to a lower dimensional space. More specifically, we assume that there exist $r, D > 0$ such that

$$r\mathcal{B} \subseteq \mathcal{K} \subseteq D\mathcal{B}. \quad (1)$$

We assume w.l.o.g. that the set contains 0, as we can always translate \mathcal{K} . We assume that each loss function ℓ_t is Lipschitz continuous, i.e., there exists a constant G such that

$$|\ell_t(x) - \ell_t(y)| \leq G\|x - y\|, \quad \forall x, y \in \mathcal{K}, \forall t. \quad (2)$$

For $\sigma \geq 0$, the function ℓ is called σ -strongly convex on the set \mathcal{K} if

$$\ell(x) \geq \ell(y) + \nabla\ell(y)^\top(x - y) + \frac{\sigma}{2}\|x - y\|^2, \quad \forall x, y \in \mathcal{K}. \quad (3)$$

The case $\sigma = 0$ merely implies convexity of ℓ . We call a function L -smooth on \mathcal{K} if it is differentiable on an open set containing \mathcal{K} and its gradient is Lipschitz continuous with a constant L , i.e.,

$$\|\nabla\ell(x) - \nabla\ell(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathcal{K}. \quad (4)$$

We use the notation \mathbb{E}_t to denote the conditional expectation conditioned on all randomness in the first $t - 1$ rounds.

2 Expected gradient descent using two queries per round

In this section, we present optimal algorithms (up to logarithmic factors) that query the loss function at two points on each round. We assume that the adversary's choice of ℓ_t can depend on all of the information available up to round $t - 1$, but does not depend on the player's random moves on round t . In the full information case, the player can follow the online gradient descent strategy (Zinkevich, 2003) and make the move

$$x_{t+1} = \Pi_{\mathcal{K}}(x_t - \eta_t \nabla\ell_t(x_t)), \quad (5)$$

Algorithm 2 Expected Gradient Descent with two queries per round

Input: Learning rates η_t , exploration parameter δ and shrinkage coefficient ξ .
Set $x_1 = 0$
for $t = 1, \dots, T$ **do**
 Pick a unit vector u_t uniformly at random.
 Observe $\ell_t(x_t + \delta u_t)$ and $\ell_t(x_t - \delta u_t)$.
 Set $\tilde{g}_t = \frac{d}{2\delta}(\ell_t(x_t + \delta u_t) - \ell_t(x_t - \delta u_t))u_t$.
 Update $x_{t+1} = \Pi_{(1-\xi)\mathcal{K}}(x_t - \eta_t \tilde{g}_t)$.
end for

where $\Pi_{\mathcal{K}}(x)$ denotes the Euclidean projection of x onto the set \mathcal{K} , and η_t is the step size or *learning rate*. In our partial information setting, we follow Flaxman et al. (2005) and replace $\nabla \ell_t(x_t)$ with an estimate \tilde{g}_t , obtained by evaluating the loss function at two random points around x_t . The player makes the move

$$x_{t+1} = \Pi_{(1-\xi)\mathcal{K}}(x_t - \eta_t \tilde{g}_t), \quad (6)$$

where $\xi \in (0, 1)$ and $(1 - \xi)\mathcal{K}$ is shorthand for $\{(1 - \xi)x : x \in \mathcal{K}\}$. The projection is made onto the shrunk set $(1 - \xi)\mathcal{K}$ to ensure that the random query points around x_{t+1} belong to \mathcal{K} . In particular, for any $x \in (1 - \xi)\mathcal{K}$ and any unit vector u it holds that $(x + \delta u) \in \mathcal{K}$ for any δ in $[0, \xi r]$ (Flaxman et al., 2005, Observation 2). Algorithm 2 gives a complete description of the Expected Gradient Descent method with two queries per round. The intuition behind this method is that any random realization of \tilde{g}_t is a good proxy for the directional derivative $\nabla \ell_t(x_t)^\top u_t$. Hence taking expectation over a random choice of u_t gives us a good estimator for $\nabla \ell_t(x_t)$.

The difference between Algorithm 2 and the algorithm proposed by Flaxman et al. (2005) is that the latter relies on a single function evaluation on each round. The single value $\ell_t(x_t + \delta u_t)$ is used to construct the gradient estimator

$$g_t = \frac{d}{\delta} \ell_t(x_t + \delta u_t) u_t. \quad (7)$$

Let v be a uniform random vector in the unit ball \mathcal{B} , and define the smoothed loss function

$$\hat{\ell}_t(x) = \mathbb{E}_v \ell_t(x + \delta v).$$

Note that $\hat{\ell}_t$ is Lipschitz continuous with the same constant G , and $\hat{\ell}_t$ is always differentiable even if ℓ_t is not. Through a clever use of Stokes' theorem, Flaxman et al. (2005) showed that g_t in (7) is a conditionally unbiased estimator of $\nabla \hat{\ell}_t(x_t)$, i.e.,

$$\mathbb{E}_t[g_t] = \nabla \hat{\ell}_t(x_t). \quad (8)$$

Their analysis uses the facts that the update of Equation (6) performs expected gradient descent on the functions $\hat{\ell}_t$ and that $\hat{\ell}_t(x_t)$ and $\ell_t(x_t)$ are close when δ is small. However, the resulting bounds on expected regret are much worse than bounds for gradient descent in the full information case. This is partly due to the large norm of the gradient estimator in (7) when δ is small.

The key insight in this section is that, for the entire class of Lipschitz continuous functions, one can use two function evaluations to construct gradient estimators that have a bounded norm, which leads to much improved regret bounds. First, we note that the gradient estimator in Algorithm 2,

$$\tilde{g}_t = \frac{d}{2\delta} (\ell_t(x_t + \delta u_t) - \ell_t(x_t - \delta u_t)) u_t, \quad (9)$$

also satisfies the unbiasedness condition (8) as the distribution of u_t is symmetric. To show that it has bounded norm, we have

$$\begin{aligned} \|\tilde{g}_t\| &= \frac{d}{2\delta} \|(\ell_t(x_t + \delta u_t) - \ell_t(x_t - \delta u_t)) u_t\| \\ &= \frac{d}{2\delta} |\ell_t(x_t + \delta u_t) - \ell_t(x_t - \delta u_t)| \\ &\leq \frac{dG}{2\delta} \|2\delta u_t\| = Gd, \end{aligned}$$

where in the inequality above we use the Lipschitz property (2).

In order to analyze the regret of Algorithm 2, we also define the functions

$$h_t(x) = \hat{\ell}_t(x) + (\tilde{g}_t - \nabla \hat{\ell}_t(x_t))^\top x, \quad \forall t.$$

It is easily seen that $\nabla h_t(x_t) = \tilde{g}_t$, and therefore $\|\nabla h_t(x_t)\| \leq Gd$ for all t . Also $\mathbb{E}_t h_t(x) = \hat{\ell}_t(x)$ for any x that is independent of u_t . Moreover, for any fixed $x, y \in \mathcal{K}$,

$$\begin{aligned} |h_t(x) - h_t(y)| &\leq |\hat{\ell}_t(x) - \hat{\ell}_t(y)| + |(\tilde{g}_t - \nabla \hat{\ell}_t(x_t))^\top (x - y)| \\ &\leq G\|x - y\| + (\|\tilde{g}_t\| + \|\nabla \hat{\ell}_t(x_t)\|)\|x - y\| \\ &\leq G(d + 2)\|x - y\| \leq 3Gd\|x - y\| \quad \text{since } \hat{\ell}_t \text{ is } G \text{ Lipschitz.} \end{aligned}$$

Therefore the Lipschitz constant of h_t is bounded by $3Gd$. In addition, the convexity of h_t follows from that of ℓ_t . With this definition, Algorithm 2 amounts to performing *deterministic* gradient descent on the convex functions h_t , with projections on the shrunk convex set $(1 - \xi)\mathcal{K}$. In the analysis below, we first state a regret bound for deterministic gradient descent on the functions h_t due to Bartlett et al. (2008b), then use it to prove the desired regret bound on the functions ℓ_t .

We assume that ℓ_t is σ_t -strongly convex (3) for some $\sigma_t \geq 0$. It follows that the functions $\hat{\ell}_t$ and h_t are also σ_t -strongly convex. Assume, without loss of generality, that $\sigma_1 > 0$. As we see later, this can always be done by adding a strongly convex regularization term, and it does not affect the overall regret bound. We use the shorthand $\sigma_{s:t}$ to denote $\sum_{\tau=s}^t \sigma_\tau$. Then $\sigma_{1:t}$ is always positive if $\sigma_1 > 0$.

We begin our analysis by stating a general upper bound on the regret of Online Gradient Descent, due to Bartlett et al. (2008b).

Lemma 1 (Bartlett et al. (2008b)) *If the Online Gradient Descent algorithm (5) is performed over a convex set \mathcal{S} on σ_t -strongly convex functions h_t with $\eta_t = 1/\sigma_{1:t}$, then for any $x \in \mathcal{S}$,*

$$\sum_{t=1}^T h_t(x_t) - \sum_{t=1}^T h_t(x) \leq \frac{d^2 G^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}}.$$

We will also require the following lemma, which relates the desired regret on the losses ℓ_t with the moves $y_{t,1} = x_t + \delta u_t$ and $y_{t,2} = x_t - \delta u_t$, to the regret on the losses $\hat{\ell}_t$ with the move x_t .

Lemma 2 *For any point $x \in \mathcal{K}$,*

$$\sum_{t=1}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \sum_{t=1}^T \ell_t(x) \leq \sum_{t=1}^T \hat{\ell}_t(x_t) - \sum_{t=1}^T \hat{\ell}_t((1 - \xi)x) + 3TG\delta + TGD\xi.$$

Proof: By the Lipschitz property (2),

$$\ell_t(y_{t,1}) = \ell_t(x_t + \delta u_t) \leq \ell_t(x_t) + G\delta\|u_t\| \quad \text{and} \quad \ell_t(y_{t,2}) = \ell_t(x_t - \delta u_t) \leq \ell_t(x_t) + G\delta\|u_t\|.$$

Since $\|u_t\| = 1$ for all t , we get

$$\frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) \leq \ell_t(x_t) + G\delta. \tag{10}$$

The Lipschitz property and the assumption $\|x\| \leq D$ also imply that, for all $x \in \mathcal{K}$,

$$\hat{\ell}_t((1 - \xi)x) \leq \hat{\ell}_t(x) + GD\xi. \tag{11}$$

We can also relate $\ell_t(x_t)$ and $\hat{\ell}_t(x_t)$ using the Lipschitz property:

$$|\ell_t(x_t) - \hat{\ell}_t(x_t)| = |\ell_t(x_t) - \mathbb{E}_t \ell_t(x_t + \delta v)| \leq \mathbb{E}_t |\ell_t(x_t) - \ell_t(x_t + \delta v)| \leq \mathbb{E}_t G\delta\|v\|,$$

where we use the fact that the random vector v in the definition of $\hat{\ell}_t(x_t)$ is independent of x_t and ℓ_t . Using the fact $\|v\| \leq 1$ gives

$$\ell_t(x_t) \leq \hat{\ell}_t(x_t) + G\delta \quad \text{and} \quad \hat{\ell}_t((1 - \xi)x) \leq \ell_t((1 - \xi)x) + G\delta \quad \forall x \in \mathcal{K}. \tag{12}$$

Combining the inequalities (10), (11) and (12), we have

$$\frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) + \hat{\ell}_t((1 - \xi)x) \leq \hat{\ell}_t(x_t) + \ell_t(x) + 3G\delta + GD\xi.$$

Finally, summing both sides of the above inequality for all t from 1 to T and rearranging terms gives the desired statement. \blacksquare

With the above lemma, it suffices to bound regret on the losses $\hat{\ell}_t$ for the sequence x_t against the set $\mathcal{K}(1 - \xi)$. We can now state a bound on the expected regret of Algorithm 2.

Theorem 3 Let assumptions (1) and (2) hold, and ℓ_t be σ_t -strongly convex where $\sigma_1 > 0$. If Algorithm 2 is run with $\eta_t = \frac{1}{\sigma_{1:t}}$, $\delta = \frac{\log T}{T}$ and $\xi = \frac{\delta}{r}$, then for any $x \in \mathcal{K}$,

$$\mathbb{E} \sum_{t=1}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \mathbb{E} \sum_{t=1}^T \ell_t(x) \leq \frac{d^2 G^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}} + G \log(T) \left(3 + \frac{D}{r} \right).$$

Proof: As pointed out earlier, since $\nabla h_t(x_t) = \tilde{g}_t$, Algorithm 2 is actually performing gradient descent (as if with full information) on the functions h_t restricted to the convex set $(1 - \xi)\mathcal{K}$. Using Lemma 1, we have that

$$\sum_{t=1}^T h_t(x_t) - \sum_{t=1}^T h_t((1 - \xi)x) \leq \frac{d^2 G^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}}.$$

Taking expectations, and using Equation (8), we conclude that

$$\mathbb{E} \sum_{t=1}^T \hat{\ell}_t(x_t) - \mathbb{E} \sum_{t=1}^T \hat{\ell}_t((1 - \xi)x) \leq \frac{d^2 G^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}}.$$

Using Lemma 2, we get that

$$\mathbb{E} \sum_{t=1}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \mathbb{E} \sum_{t=1}^T \ell_t(x) \leq \frac{d^2 G^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}} + 3TG\delta + TGD\xi.$$

Plugging in the stated values of δ and ξ completes the proof. \blacksquare

We can take δ arbitrarily close to 0, but that doesn't improve the bound beyond constant factors. With Theorem 3 handy, we are all set to prove the following corollaries.

Corollary 4 If assumptions (1) and (2) hold and Algorithm 2 is run with $\eta_t = \frac{1}{\sqrt{T}}$, $\delta = \frac{\log T}{T}$ and $\xi = \frac{\delta}{r}$, then

$$\mathbb{E} \sum_{t=1}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \min_{x \in \mathcal{K}} \mathbb{E} \sum_{t=1}^T \ell_t(x) \leq (d^2 G^2 + D^2) \sqrt{T} + G \log(T) \left(3 + \frac{D}{r} \right).$$

Proof: Since we do not assume that the loss functions ℓ_t , for $t \geq 1$, are strongly convex, we add a fictitious round in the beginning with $\ell_0(x) = \frac{\sqrt{T}}{2} \|x\|^2$, which has $\sigma_0 = \sqrt{T}$. Setting $\sigma_t = 0$ for $t \geq 1$ leads to $\sigma_{0:t} = \sqrt{T}$ for all $t \geq 0$. Plugging this in the result of Theorem 3, we have for all $x \in \mathcal{K}$,

$$\mathbb{E} \sum_{t=0}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \mathbb{E} \sum_{t=0}^T \ell_t(x) \leq \frac{d^2 G^2}{2} \sum_{t=0}^T \frac{1}{\sqrt{T}} + G \log(T) \left(3 + \frac{D}{r} \right). \quad (13)$$

For the regret in the initial fictitious round, we have

$$\frac{1}{2} (\ell_0(y_{0,1}) + \ell_0(y_{0,2})) - \ell_0(x) = \frac{1}{2} \left(\frac{\sqrt{T}}{2} \|y_{0,1}\|^2 + \frac{\sqrt{T}}{2} \|y_{0,2}\|^2 \right) - \frac{\sqrt{T}}{2} \|x\|^2 \geq -\frac{\sqrt{T}}{2} D^2.$$

Rearranging the inequality (13) gives

$$\begin{aligned} \mathbb{E} \sum_{t=1}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \mathbb{E} \sum_{t=1}^T \ell_t(x) &\leq \frac{d^2 G^2}{2} \frac{T+1}{\sqrt{T}} + \frac{\sqrt{T}}{2} D^2 + G \log(T) \left(3 + \frac{D}{r} \right) \\ &\leq (d^2 G^2 + D^2) \sqrt{T} + G \log(T) \left(3 + \frac{D}{r} \right). \end{aligned}$$

Since this holds for all $x \in \mathcal{K}$, it is certainly true for $\arg \min_{x \in \mathcal{K}} \mathbb{E} \sum_{t=1}^T \ell_t(x)$. \blacksquare

Corollary 5 Suppose that assumptions (1) and (2) hold, and each loss function ℓ_t is σ -strongly convex for some $\sigma > 0$. If Algorithm 2 is run with $\eta_t = \frac{1}{\sigma t}$, $\delta = \frac{\log T}{T}$ and $\xi = \frac{\delta}{r}$, then

$$\mathbb{E} \sum_{t=1}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \min_{x \in \mathcal{K}} \mathbb{E} \sum_{t=1}^T \ell_t(x) \leq G \log(T) \left(\frac{d^2 G}{\sigma} + 3 + \frac{D}{r} \right).$$

Proof: In this case, $\sigma_t = \sigma > 0$ and therefore $\sigma_{1:t} = \sigma t$. Hence we can directly plug these values into Theorem 3 to conclude

$$\begin{aligned} \mathbb{E} \sum_{t=1}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \min_{x \in \mathcal{K}} \mathbb{E} \sum_{t=1}^T \ell_t(x) &\leq \frac{d^2 G^2}{2} \sum_{t=1}^T \frac{1}{\sigma t} + G \log(T) \left(3 + \frac{D}{r} \right) \\ &\leq \frac{d^2 G^2}{2\sigma} (1 + \log(T)) + G \log(T) \left(3 + \frac{D}{r} \right). \\ &\leq \frac{d^2 G^2}{\sigma} \log(T) + G \log(T) \left(3 + \frac{D}{r} \right) \quad \text{for } T \geq 3. \end{aligned}$$

■

While expected regret bounds are interesting, the regret may still have a high variance. In order to argue that Algorithm 2 often enjoys a small regret, we need to prove a bound that holds with high probability. For that, we turn to concentration inequalities for martingales. For any point $x \in (1 - \xi)\mathcal{K}$, define

$$Z_t = \hat{\ell}_t(x_t) - \hat{\ell}_t(x) - h_t(x_t) + h_t(x).$$

Since $\mathbb{E}_t h_t(x) = \hat{\ell}_t(x)$ for any x that is independent of u_t , we have $\mathbb{E}_t Z_t = 0$. In addition,

$$|Z_t| \leq |\hat{\ell}_t(x_t) - \hat{\ell}_t(x)| + |h_t(x_t) - h_t(x)| \leq G \|x_t - x\| + 3Gd \|x_t - x\| \leq 8GdD.$$

Thus the sequence Z_t is a bounded martingale difference sequence. Hence, we can use the Hoeffding-Azuma inequality to derive a high probability guarantee for the case of convex, Lipschitz functions.

Theorem 6 *Suppose assumptions (1) and (2) hold. If Algorithm 2 is run with $\eta_t = \frac{1}{\sqrt{T}}$, $\delta = \frac{\log T}{T}$ and $\xi = \frac{\delta}{r}$. Then for any fixed $x \in \mathcal{K}$ and any $\delta_1 > 0$, with probability at least $1 - \delta_1$,*

$$\sum_{t=1}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \sum_{t=1}^T \ell_t(x) \leq (d^2 G^2 + D^2) \sqrt{T} + G \log(T) \left(3 + \frac{D}{r} \right) + 8dGD \sqrt{2T \log(1/\delta_1)}.$$

Proof: Using the Hoeffding-Azuma inequality, we conclude that

$$\mathbb{P} \left(\sum_{t=1}^T Z_t > \epsilon \right) \leq \exp \left(-\frac{\epsilon^2}{2TB^2} \right),$$

where $B = 8GdD$ as shown before. Let $\delta_1 = \exp \left(-\frac{\epsilon^2}{2TB^2} \right)$. Then $\epsilon = B \sqrt{2T \log(1/\delta_1)}$. Hence we know that with probability at least $1 - \delta_1$

$$\sum_{t=1}^T \hat{\ell}_t(x_t) - \hat{\ell}_t(x(1 - \xi)) \leq \sum_{t=1}^T h_t(x_t) - h_t(x(1 - \xi)) + B \sqrt{2T \log(1/\delta_1)}.$$

Now using Lemma 2,

$$\sum_{t=1}^T \frac{1}{2} (\ell_t(y_{t,1}) + \ell_t(y_{t,2})) - \sum_{t=1}^T \ell_t(x) \leq \sum_{t=1}^T h_t(x_t) - h_t(x(1 - \xi)) + B \sqrt{2T \log(1/\delta_1)} + 3TG\delta + TGD\xi.$$

To bound the regret $\sum_{t=1}^T h_t(x_t) - h_t(x(1 - \xi))$, we use Lemma 1 on the regularized loss sequence exactly like in the proof of Corollary 4. Plugging in the stated values of η_t and δ proves the theorem.

■

When the adversary is adaptive, however, the loss sequence depends on the player's moves. As a result, the best comparator, which minimizes $\sum_{t=1}^T \ell_t(x)$, depends on the player's moves too. The high probability result stated above doesn't allow us to compete with such a comparator, as the comparator x in the theorem statement is fixed ahead of time and is not allowed to depend on the random x_t 's. To obtain a regret bound against $\min_{x \in \mathcal{K}} \sum_{t=1}^T \ell_t(x)$, we need to take the high probability bound for a fixed $x \in \mathcal{K}$ described above, and combine it with an appropriate union bound over the entire set to account for all possible minimizers of the cumulative loss sequence. This union bound can be taken either over a barycentric spanner, or over a cover of size at most

$(4dT)^{d/2}$. This step is the same as the one described in Bartlett et al. (2008a) or Dani and Hayes (2006), and is not repeated here.

While this result provides nearly tight (up to log factors and constants) guarantees on high probability regret for general convex functions, it falls short for the case of strongly convex functions. We note that in applying Hoeffding-Azuma's concentration inequality, we incur an additional term of $\tilde{O}(\sqrt{T})$. Showing an $O(\log T)$ regret bound for strongly convex functions that holds with high probability remains an open question. We would like to mention that the techniques of Kakade and Tewari (2009), which provide such a result in the online to batch conversion setting, are not applicable to our case. This is because in stochastic optimization of strongly convex functions, the sequence x_t converges rapidly to the optimal point. This rapid concentration doesn't occur in an adversarial bandit optimization setting; the sequence x_t might actually track the oscillations of the losses ℓ_t and give a much lower regret than any constant point. Such cases present a novel difficulty and resolving this difficulty is an interesting question for future research.

3 A general class of gradient estimators for smooth functions

In the previous section, we focused on a gradient estimator based on two function evaluations along a random direction uniformly distributed on the unit sphere centered at x_t . It is natural to ask if one can obtain similar regret bounds for other estimators based on random sampling with different probability distributions. It turns out that with an additional smoothness assumption on the loss functions, we can indeed show similar regret bounds with more general gradient estimators.

The additional smoothness assumption is that each loss function ℓ_t is L -smooth, i.e., they satisfy the condition (4). We recall that a direct consequence of (4) is the following second-order property

$$\ell_t(x) \leq \ell_t(y) + \langle \nabla \ell_t(y), x - y \rangle + \frac{L}{2} \|x - y\|^2, \quad \forall x, y \in \mathcal{K}. \quad (14)$$

If the loss functions are L -smooth, then we can show that the expectation of the gradient estimator in Equation (9) is close to the true gradient. We start with a useful fact from linear algebra.

Lemma 7 *For any fixed $v \in \mathbb{R}^d$ and for $u \in \mathbb{R}^d$ chosen randomly with $\|u\| = 1$, it holds that $v = d\mathbb{E} \langle v, u \rangle u$.*

Proof: The lemma is based on the observation that $\mathbb{E}uu^\top = \frac{1}{d}I$, where u is a random unit vector and I is the $d \times d$ identity matrix. To see this, recall that symmetry implies $\mathbb{E}u_i u_j = 0$ for any two coordinates i, j . Also u is a unit vector so $\mathbb{E} \sum_i u_i^2 = 1$. Again by symmetry, this implies that $\mathbb{E}u_i^2 = \frac{1}{d}$ for all i . Therefore, $Iv = (d\mathbb{E}uu^\top)v$, which equals $d\mathbb{E} \langle v, u \rangle u$. ■

The above lemma allows us to conclude

$$\begin{aligned} \|\mathbb{E}_t \tilde{g}_t - \nabla \ell_t(x_t)\| &= \|\mathbb{E}_t \tilde{g}_t - d\mathbb{E}_t \langle \nabla \ell_t(x_t), u_t \rangle u_t\| \leq \mathbb{E}_t \|\tilde{g}_t - d \langle \nabla \ell_t(x_t), u_t \rangle u_t\| \\ &\leq d\mathbb{E}_t \left| \frac{1}{2\delta} (\ell_t(x_t + \delta u_t) - \ell_t(x_t - \delta u_t)) - \langle \nabla \ell_t(x_t), u_t \rangle \right| \quad \text{since } \|u_t\| = 1. \end{aligned}$$

By the smoothness assumption, more specifically (14), we have

$$\ell_t(x_t + \delta u_t) \leq \ell_t(x_t) + \delta \langle \nabla \ell_t(x_t), u_t \rangle + \frac{L\delta^2}{2}.$$

Also by convexity of ℓ_t , $\ell_t(x_t - \delta u_t) \geq \ell_t(x_t) - \delta \langle \nabla \ell_t(x_t), u_t \rangle$. Combining the two inequalities above, we get

$$\frac{1}{2\delta} (\ell_t(x_t + \delta u_t) - \ell_t(x_t - \delta u_t)) - \langle \nabla \ell_t(x_t), u_t \rangle \leq \frac{L\delta}{4}.$$

Similarly by using (14) on $\ell_t(x_t - \delta u_t)$ and the convexity inequality on $\ell_t(x_t + \delta u_t)$, we can lower bound the left-hand side of the above inequality by $-\frac{L\delta}{4}$. This allows us to obtain

$$\|\mathbb{E}_t \tilde{g}_t - \nabla \ell_t(x_t)\| \leq \frac{dL\delta}{4}.$$

It turns out that the boundedness of the gradient estimator \tilde{g}_t along with the closeness of its expectation to the true gradient is sufficient to reproduce the regret bounds of the previous section. In particular, we do not need to rely on random vectors uniformly distributed on the unit sphere centered at x_t and we do not need to use the function $\hat{\ell}_t(x)$. To illustrate this point, we only show how a corresponding version of Theorem 3 is proved in this general case. The corresponding versions of other results follow in a similar fashion.

Theorem 8 Assume that (1) and (2) hold, each function ℓ_t is σ_t -strongly convex, and $\sigma_1 > 0$. Suppose that on round t the player issues k random queries $y_{t,1}, \dots, y_{t,k}$, constructs a gradient estimator \tilde{g}_t , and uses the algorithm $x_{t+1} = \Pi_{\mathcal{K}(1-\xi)}(x_t - \eta_t \tilde{g}_t)$ with $\eta_t = \frac{1}{\sigma_{1:t}}$, $\delta = \frac{\log T}{T}$ and $\xi = \frac{\delta}{r}$. If the gradient estimator satisfies the following conditions for all $t \geq 1$:

- (i) $\|x_t - y_{t,i}\| \leq \delta$ for $i = 1, \dots, k$.
- (ii) $\|\tilde{g}_t\| \leq G_1$ for some constant G_1 .
- (iii) $\|\mathbb{E}_t \tilde{g}_t - \nabla \ell_t(x_t)\| \leq c\delta$ for some constant c .

Then for any fixed $x \in \mathcal{K}$ we have

$$\mathbb{E} \sum_{t=1}^T \frac{1}{k} \sum_{i=1}^k \ell_t(y_{t,i}) - \mathbb{E} \sum_{t=1}^T \ell_t(x) \leq \frac{G_1^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}} + G \log(T) \left(1 + 2c + \frac{D}{r}\right).$$

Proof: We start by defining $h_t(x) = \ell_t(x) + (\tilde{g}_t - \nabla \ell_t(x))^\top x$. Then it is clear that h_t has the same convexity properties as ℓ_t and $\nabla h_t(x_t) = \tilde{g}_t$. So Lemma 1 still holds, with a gradient bound of G_1 instead of dG , and we get

$$\sum_{t=1}^T h_t(x_t) - h_t(x) \leq \frac{G_1^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}}.$$

Then we can take expectations to conclude

$$\begin{aligned} \mathbb{E} \sum_{t=1}^T [\ell_t(x_t) - \ell_t(x)] &= \mathbb{E} \sum_{t=1}^T [h_t(x_t) - h_t(x)] + \mathbb{E} \sum_{t=1}^T [\ell_t(x_t) - h_t(x_t) - \ell_t(x) + h_t(x)] \\ &\leq \frac{G_1^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}} + \mathbb{E} \sum_{t=1}^T (\mathbb{E}_t \tilde{g}_t - \nabla \ell_t(x_t))^\top (x_t - x) \\ &\leq \frac{G_1^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}} + 2c\delta DT. \end{aligned}$$

In the first inequality above, we used convexity of ℓ_t and h_t . We can now use arguments similar to the proof of Lemma 2 to conclude

$$\mathbb{E} \sum_{t=1}^T \frac{1}{k} \sum_{i=1}^k \ell_t(y_{t,i}) - \mathbb{E} \sum_{t=1}^T \ell_t(x) \leq \frac{G_1^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}} + 2cD\delta T + GT\delta + GDT\xi.$$

Plugging in the values of δ and ξ proves the theorem. \blacksquare

As shown in the beginning of this section, the assumptions of Theorem 8 are satisfied with the gradient estimator (9) if the loss functions ℓ_t are L -smooth. In that case, we have $G_1 = dG$ and $c = dL/4$, and $k = 2$ queries per round suffice.

Theorem 8 also generalizes the special case of optimizing with noisy gradients, which has been previously studied in the optimization community. Suppose that for each query point x_t , the player only has access to a noisy version of the gradient \tilde{g}_t . In this case, we have $k = 1$ and $y_{t,1} = x_t$, and it is not even necessary to assume the smoothness condition. If the approximate gradient satisfies the conditions (ii) and (iii) in Theorem 8, then we have the stated regret bound in expectation. If the approximate gradient has a deterministic bound, i.e., $\|\tilde{g}_t - \nabla \ell_t(x_t)\| \leq c\delta$ for all t , then the regret bound becomes deterministic.

As a more interesting application of this general result, we demonstrate another estimator \tilde{g}_t that meets the conditions of Theorem 8. On each round t , the player picks an integer i_t from the set $\{1, \dots, d\}$ uniformly at random, queries the function values at two points along the corresponding coordinate axes e_{i_t} ; $y_{t,1} = x_t + \delta e_{i_t}$, $y_{t,2} = x_t - \delta e_{i_t}$. Then it constructs the estimator

$$\tilde{g}_t = \frac{d}{2\delta} (\ell_t(y_{t,1}) - \ell_t(y_{t,2})) e_{i_t}.$$

It is clear that $\|x_t - y_{t,i}\| \leq \delta$. As in the previous section, this gradient estimator has a bounded norm with $G_1 = dG$. Moreover,

$$\mathbb{E}_t \tilde{g}_t = \frac{1}{2\delta} \sum_{i=1}^d (\ell_t(x_t + \delta e_i) - \ell_t(x_t - \delta e_i)) e_i.$$

Algorithm 3 Gradient descent with deterministic estimator based on $d + 1$ points

Input: Learning rates η_t , exploration parameter δ and shrinkage coefficient ξ .
Set $x_1 = 0$
for $t = 1, \dots, T$ **do**
 Observe $\ell_t(x_t), \ell_t(x_t + \delta e_i)$ for $i = 1, \dots, d$.
 Set $\tilde{g}_t = \frac{1}{\delta} \sum_{i=1}^d (\ell_t(x_t + \delta e_i) - \ell_t(x_t)) e_i$.
 Update $x_{t+1} = \Pi_{(1-\xi)\mathcal{K}}(x_t - \eta_t \tilde{g}_t)$.
end for

Then

$$\begin{aligned} \|\mathbb{E}_t \tilde{g}_t - \nabla \ell_t(x_t)\| &= \sqrt{\sum_{i=1}^d \left| \frac{1}{2\delta} (\ell_t(x_t + \delta e_i) - \ell_t(x_t - \delta e_i)) - \langle \nabla \ell_t(x_t), e_i \rangle \right|^2} \\ &\leq \frac{\sqrt{d}L\delta}{4}. \end{aligned}$$

Thus Theorem 8 applies to this estimator with $G_1 = dG$ and $c = \sqrt{d}L/4$. This means that a coordinate descent style algorithm that uses only 1-dimensional gradient estimators will also exhibit a low regret for optimizing smooth convex Lipschitz functions.

Since the gradient estimator is 1-dimensional along a coordinate axis, gradient updates become computationally more efficient. We also note that this improvement in per-iteration cost comes at no worsening of the regret bound when compared to using a random direction on the unit sphere. In the full information case, stochastic coordinate descent approaches usually have a convergence rate slower by a factor of d compared to gradient descent, which offsets the computational gains (Shalev-Shwartz & Tewari, 2009; Tseng & Yun, 2009). In this case, however, both the gradient descent method and coordinate descent have to rely on an estimator with the same amount of variance, which leads to similar regret guarantees in the two cases and gives the coordinate descent approach a clear computational edge. This example is just a hint of the strength of Theorem 8.

4 Deterministic algorithms for completely adaptive adversaries

In the previous sections, we considered adversaries that know the player's past moves, but do not know the player's current random move. This seems reasonable for a randomized algorithm, because if the adversary can see the player's random bits, then randomization is futile. However, in some cases, the online interaction mandates an adversarial response dependent not only on the player's previous moves, but also on the current move. That is, after the player plays a point x_t , the loss function ℓ_t is chosen with the knowledge of x_t . While such a scenario is readily tackled in the full information setting with standard gradient based algorithms, it is impossible to adapt to such feedback in the bandit setting, as explained in Section 1. This prompts the question if there is any partial feedback setting where we might be able to effectively compete against such an adversary. In this section, we show one special partial feedback scenario where it is possible to compete against such a *completely adaptive* adversary.

We assume that the player is allowed to query each loss function at up to $d + 1$ points. In this case, the player can play the points $x_t, x_t + \delta e_i$ for $i = 1, \dots, d$, where e_i are the standard unit basis vectors, and then construct a *deterministic* gradient approximation

$$\tilde{g}_t = \frac{1}{\delta} \sum_{i=1}^d (\ell_t(x_t + \delta e_i) - \ell_t(x_t)) e_i. \quad (15)$$

The details of the player's moves are described in Algorithm 3.

We assume that each loss function ℓ_t is L -smooth and σ_t -strongly convex on \mathcal{K} . Such functions are always Lipschitz continuous, and we can derive a bound on the Lipschitz constant based on L and D . However, for convenience, we simply assume (2) holds with the constant G . We can derive just as in previous sections that

$$\|\tilde{g}_t\| \leq dG, \quad \|\tilde{g}_t - \nabla \ell_t(x_t)\| \leq \frac{\sqrt{d}L\delta}{2}. \quad (16)$$

These properties of \tilde{g}_t are the deterministic version of the properties discussed in the previous section for randomized algorithms. They immediately give us partial feedback algorithms based on using approximate gradients in the corresponding full information analogues. For instance, we can show the same regret guarantee against a stronger class of adversaries.

Theorem 9 Let $\{\ell_t\}_{t=1}^T$ be convex functions chosen by a completely adaptive adversary. Suppose the conditions (1) and (2) hold. In addition, let each ℓ_t be σ_t strongly convex and L -smooth. If Algorithm 3 is run with $\eta_t = \frac{1}{\sigma_{1:t}}$, $\delta = \frac{\log T}{T}$ and $\xi = \frac{\delta}{r}$, then

$$\sum_{t=1}^T \frac{1}{d+1} \left(\ell_t(x_t) + \sum_{i=1}^d \ell_t(x_t + \delta e_i) \right) - \min_{x \in \mathcal{K}} \sum_{t=1}^T \ell_t(x) \leq \frac{d^2 G^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}} + G \log(T) \left(1 + \frac{\sqrt{d} L \delta}{2} + \frac{D}{r} \right).$$

Proof: The proof essentially mimics that of Theorem 8. We define $h_t(x) = \ell_t(x) + (\tilde{g}_t - \nabla \ell_t(x_t))^\top x$. Then $\|\nabla h_t(x_t)\| = \|\tilde{g}_t\| \leq dG$. So we obtain from Lemma 1 for any $x \in (1 - \xi)\mathcal{K}$,

$$\sum_{t=1}^T h_t(x_t) - h_t(x) \leq \frac{d^2 G^2}{2} \sum_{t=1}^T \frac{1}{\sigma_{1:t}}.$$

Proceeding as in the proof of Theorem 8 and using $\|\tilde{g}_t - \nabla \ell_t(x_t)\| \leq \frac{\sqrt{d} L \delta}{2}$ gives the result. \blacksquare

It turns out that we can say more about this deterministic gradient estimator. More specifically, we can use it to develop quasi-Newton type methods that can achieve a sharper regret bound for the class of exp-concave functions. Recall that a loss function ℓ_t is α -exp-concave if $\exp(-\alpha \ell_t(x))$ is a concave function. For example, if a function has a Lipschitz constant G and is σ -strongly convex, then the exp-concave property holds with $\alpha = G^2/\sigma$. However, the exp-concave condition is weaker than assuming the bounds G and σ . These functions arise in applications like portfolio optimization, squared-error regression etc.

All the algorithms discussed until now are gradient-descent based. Gradient-descent methods are often insufficient to obtain the optimal regret for the general class of exp-concave functions. Hazan et al. (2007) developed an online Newton-step algorithm that achieves an optimal $O(\log T)$ regret on these functions. In order to implement approximate second-order methods using gradient estimators, we also need to control error in second order gradient terms. To demonstrate good second-order properties of the gradient estimator (15), we use the following fact from linear algebra.

Lemma 10 Let u and v be two vectors in \mathbb{R}^d such that $\|u - v\| \leq \theta$. Then for any x

$$x^\top (uu^\top - vv^\top)x \leq \|x\|^2 \theta (2\|v\| + \theta).$$

Proof: We have

$$\begin{aligned} x^\top (uu^\top - vv^\top)x &= (x^\top u)^2 - (x^\top v)^2 = x^\top (u + v)x^\top (u - v) \\ &\leq \|x\|^2 \|u + v\| \|u - v\| \leq \|x\|^2 \theta (2\|v\| + \theta), \end{aligned}$$

where in the last inequality we used $\|u + v\| = \|2v + (u - v)\| \leq 2\|v\| + \|u - v\| \leq 2\|v\| + \theta$. \blacksquare

We apply this lemma with $u = \tilde{g}_t$ and $v = \nabla \ell_t(x_t)$ to conclude that for any vector x

$$x^\top (\tilde{g}_t^\top \tilde{g}_t - \nabla \ell_t(x_t)^\top \nabla \ell_t(x_t))x \leq \|x\|^2 \frac{\sqrt{d} L \delta}{2} \left(2G + \frac{\sqrt{d} L \delta}{2} \right). \quad (17)$$

With this second-order approximation bound in mind, we present the Bandit Online Newton-Step algorithm (BONES). This algorithm is a bandit version of the Online Newton Step algorithm, presented in Hazan et al. (2007).

We begin the analysis of the BONES algorithm by first proving the following lemma.

Lemma 11 Let \tilde{g}_t and β be as defined in Algorithm 4. Then for any $x \in (1 - \xi)\mathcal{K}$,

$$\sum_{t=1}^T \tilde{g}_t^\top (x_t - x) - \frac{\beta}{2} (x - x_t)^\top \tilde{g}_t \tilde{g}_t^\top (x - x_t) \leq \frac{d}{2\beta} \log(TG^2 d^4 \beta^2 + 1) + \frac{1}{2\beta}.$$

Proof: This result is based on the observation that the updates of BONES are equivalent to performing online Newton-step algorithm of Hazan et al. (2007) on the functions

$$\tilde{g}_t^\top (x - x_t) - \frac{\beta}{2} (x - x_t)^\top A_t (x - x_t).$$

Algorithm 4 Bandit Online NEWton Step (BONES) algorithm

Set $x_1 = 0, A_0 = \epsilon I_d, \beta = \frac{1}{2} \min \left\{ \frac{1}{4GD}, \alpha \right\}$.
for $t = 1, \dots, T$ **do**
 Observe $\ell_t(x_t), \ell_t(x_t + \delta e_i)$ for $i = 1, \dots, d$.
 $\tilde{g}_t = \frac{1}{\delta} \sum_{i=1}^d (\ell_t(x_t + \delta e_i) - \ell_t(x_t)) e_i$.
 $A_t = A_{t-1} + \tilde{g}_t \tilde{g}_t^\top$.
 $x_{t+1} = \Pi_{(1-\xi)\mathcal{K}}^{A_t} \left(x_t - \frac{1}{\beta} A_t^{-1} \tilde{g}_t \right)$, where $\Pi_{\mathcal{S}}^{A_t}(y) = \arg \min_{x \in \mathcal{S}} (y - x)^\top A_t (y - x)$.
end for

Following the proof of Theorem 2 in (Hazan et al., 2007), we conclude that for any $x \in (1 - \xi)\mathcal{K}$,

$$\sum_{t=1}^T \tilde{g}_t^\top (x - x_t) - \frac{\beta}{2} (x - x_t)^\top A_t (x - x_t) \leq \frac{1}{2\beta} \sum_{t=1}^T \tilde{g}_t^\top A_t^{-1} \tilde{g}_t + \frac{1}{2\beta}.$$

Further following their analysis, we note that the first summand on the right-hand side can be bounded by controlling the eigenvalues of the matrices A_t . Together with $\|\tilde{g}_t\| \leq Gd$, this gives

$$\sum_{t=1}^T \tilde{g}_t^\top (x - x_t) - \frac{\beta}{2} (x - x_t)^\top A_t (x - x_t) \leq \frac{d}{2\beta} \log \left(\frac{(Gd)^2 T}{\epsilon} + 1 \right) + \frac{1}{2\beta}.$$

Finally setting $\epsilon = \frac{1}{\beta^2 d^2}$ gives the result. \blacksquare

This lemma now allows us to prove a bound on the regret incurred by the BONES algorithm against completely adaptive adversaries.

Theorem 12 *Let $\{\ell_t\}_{t=1}^T$ be chosen by a completely adaptive adversary. Suppose the conditions (1) and (2) hold. In addition, let each ℓ_t be α -exp-concave and L -smooth. If the BONES Algorithm is run with $\delta = \frac{\log T}{T}$ and $\xi = \frac{\delta}{r}$, then*

$$\begin{aligned} \sum_{t=1}^T \frac{1}{d+1} \left(\ell_t(x_t) + \sum_{i=1}^d \ell_t(x_t + \delta e_i) \right) - \min_{x \in \mathcal{K}} \sum_{t=1}^T \ell_t(x) &\leq 4d \left(GD + \frac{1}{\alpha} \right) \log \left(1 + \frac{Td^4 D^2}{64} \right) \\ &\quad + \log(T) \left(\frac{DL\sqrt{d}}{2} + \frac{2GD}{r} \right) + o(\log(T)). \end{aligned}$$

Proof: We will only outline the main steps in the proof. The omitted details can be found in (Hazan et al., 2007). We start by recalling a second-order property of exp-concave functions (Hazan et al., 2007, Lemma 3):

$$\ell_t(x) \geq \ell_t(x_t) + \nabla \ell_t(x_t)^\top (x - x_t) + \frac{\beta}{2} (x - x_t)^\top \nabla \ell_t(x_t) \nabla \ell_t(x_t)^\top (x - x_t).$$

In particular, this observation allows us to relate regret on the function ℓ_t with that of a local quadratic approximation.

$$\begin{aligned} \sum_{t=1}^T \ell_t(x_t) - \ell_t(x) &\leq \sum_{t=1}^T \nabla \ell_t(x_t)^\top (x_t - x) - \frac{\beta}{2} (x - x_t)^\top \nabla \ell_t(x_t) \nabla \ell_t(x_t)^\top (x - x_t) \\ &\leq \sum_{t=1}^T \tilde{g}_t(x_t)^\top (x_t - x) - \frac{\beta}{2} (x - x_t)^\top \tilde{g}_t \tilde{g}_t^\top (x - x_t) \\ &\quad + \sum_{t=1}^T \|\tilde{g}_t - \nabla \ell_t(x_t)\| \|x_t - x\| + \sum_{t=1}^T \frac{\beta}{2} (x_t - x) (\tilde{g}_t \tilde{g}_t^\top - \nabla \ell_t(x_t) \nabla \ell_t(x_t)^\top) (x - x_t) \end{aligned}$$

Now the first two terms on the right-hand side are bounded in Lemma 11. For the remaining terms, we use the equations (16) and (17). As before, we also need to relate this regret to the average regret on the $y_{t,i}$'s against a point in \mathcal{K} rather than $\mathcal{K}(1 - \xi)$. These can be done as before by appealing to Lemma 2. Plugging in all the constants gives the result. \blacksquare

5 Discussion

This paper introduces the multi-point bandit feedback setting for online convex optimization under partial information. While this setting is just a slight generalization of the bandit setting, we are able to show certain sharp phase transitions based on the number of queries per round, both in the nature of the bounds and in the nature of the adversaries against which these bounds hold. We provide optimal algorithms at several points along this continuum.

For future research, it will be interesting to investigate more general partial feedback models, for instance allowing an adaptive choice of the number of queries at every round. Proving an $O(\log(T))$ regret bound with high probability for the case of strongly convex functions also remains an open question for $k = 2$ queries.

References

- Abernethy, J., Agarwal, A., Rakhlin, A., & Bartlett, P. L. (2009). A stochastic view of optimal regret through minimax duality. *Proceedings of the 22nd Annual Conference on Learning Theory*.
- Abernethy, J., & Rakhlin, A. (2009). Beating the adaptive bandit with high probability. *Proceedings of the 22nd Annual Conference on Learning Theory*.
- Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2003). The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32, 48–77.
- Bartlett, P. L., Dani, V., Hayes, T., Kakade, S., Rakhlin, A., & Tewari, A. (2008a). High-probability regret bounds for bandit online linear optimization. *Proc. of the 21st Annual COLT*.
- Bartlett, P. L., Hazan, E., & Rakhlin, A. (2008b). Adaptive online gradient descent. *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press.
- Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning and games*. Cambridge University Press.
- Conn, A. R., Scheinberg, K., & Vicente, L. N. (2009). *Introduction to derivative-free optimization*. Philadelphia, PA: SIAM.
- Dani, V., & Hayes, T. P. (2006). Robbing the bandit: Less regret in online geometric optimization against an adaptive adversary. *ACM-SIAM Symposium on Discrete Algorithms*.
- Dani, V., Hayes, T. P., & Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback. *Proceedings of the 21st Annual Conference on Learning Theory*.
- Flaxman, A. D., Kalai, A. T., & McMahan, B. H. (2005). Online convex optimization in the bandit setting: gradient descent without a gradient. *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 385–394).
- Hazan, E., Agarwal, A., & Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69, 169–192.
- Kakade, S. M., & Tewari, A. (2009). On the generalization ability of online strongly convex programming algorithms. *Advances in Neural Information Processing Systems 21* (pp. 801–808).
- Polyak, B. T., & Tsytkin, Y. Z. (1973). Pseudogradient adaption and training algorithms. *Automation and Remote Control*, 34, 377–397.
- Shalev-Shwartz, S., & Tewari, A. (2009). Stochastic methods for ℓ_1 regularized loss minimization. *Proceedings of the 26th International Conference on Machine Learning*.
- Spall, J. C. (2003). *Introduction to stochastic search and optimization: Estimation, simulation, and control*. Hoboken, NJ: John Wiley and Sons.
- Tseng, P., & Yun, S. (2009). A block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *J. of Optimization Theory and Applications*, 140, 513–535.
- Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the 20th International Conference on Machine Learning* (pp. 928–936).

Best Arm Identification in Multi-Armed Bandits

Jean-Yves Audibert
Imagine, Université Paris Est
&
Willow, CNRS/ENS/INRIA, Paris, France
audibert@imagine.enpc.fr

Sébastien Bubeck, Rémi Munos
SequeL Project, INRIA Lille
40 avenue Halley,
59650 Villeneuve d'Ascq, France
{sebastien.bubeck, remi.munos}@inria.fr

Abstract

We consider the problem of finding the best arm in a stochastic multi-armed bandit game. The regret of a forecaster is here defined by the gap between the mean reward of the optimal arm and the mean reward of the ultimately chosen arm. We propose a highly exploring UCB policy and a new algorithm based on successive rejects. We show that these algorithms are essentially optimal since their regret decreases exponentially at a rate which is, up to a logarithmic factor, the best possible. However, while the UCB policy needs the tuning of a parameter depending on the unobservable hardness of the task, the successive rejects policy benefits from being parameter-free, and also independent of the scaling of the rewards. As a by-product of our analysis, we show that identifying the best arm (when it is unique) requires a number of samples of order (up to a $\log(K)$ factor) $\sum_i 1/\Delta_i^2$, where the sum is on the suboptimal arms and Δ_i represents the difference between the mean reward of the best arm and the one of arm i . This generalizes the well-known fact that one needs of order of $1/\Delta^2$ samples to differentiate the means of two distributions with gap Δ .

1 Introduction

In the multi-armed bandit problem, at each stage, an agent (or forecaster) chooses one action (or arm), and receives a reward from it. In its stochastic version, the reward is drawn from a fixed probability distribution given the arm. The usual goal is to maximize the cumulative sum of rewards, see Robbins (1952); Auer et al. (2002) among many others. Since the forecaster does not know the distributions, he needs to explore (try) the different actions and yet, exploit (concentrate its draws on) the seemingly most rewarding arms. In this paper, we adopt a different viewpoint. We assume that after a given number of pulls, the forecaster is asked to output a recommended arm. He is then *only* evaluated by the average payoff of his recommended arm. This is the so-called pure exploration problem, Bubeck et al. (2009).

The distinguishing feature from the classical multi-armed bandit problem described above is that the exploration phase and the evaluation phase are separated. Thus, there is no explicit trade-off between the exploration and the exploitation while pulling the arms. The target of Hoeffding and Bernstein races, see Maron and Moore (1993); Mnih et al. (2008) among others, is more similar to ours. However, instead of trying to extract from a fixed number of rounds the best action, racing algorithms try to identify the best action at a given confidence level while consuming the minimal number of pulls. They optimize the budget for a given confidence level, instead of optimizing the quality of the recommendation for a given budget size. Another variant of the best arm identification task is the problem of minimal sampling times required to identify an ε -optimal arm, see in particular Domingo et al. (2002) and Even-Dar et al. (2006).

We now illustrate why this is a natural framework for numerous applications. Historically, the first occurrence of multi-armed bandit problems was given by medical trials, see Robbins (1952). In the case of a severe disease, ill patients only are included in the trial and the cost of picking the wrong treatment is high. It is important to minimize the cumulative regret, since the test and cure phases coincide. However, for cosmetic products, there exists a test phase separated from the commercialization phase, and one aims at minimizing the regret of the commercialized product rather than the cumulative regret in the test phase, which is irrelevant.

Another motivating example concerns channel allocation for mobile phone communications. During a very short time before the communication starts, a cellphone can explore the set of channels to find the best one to operate. Each evaluation of a channel is noisy and there is a limited number of evaluations

<p>Parameters available to the forecaster: the number of rounds n and the number of arms K.</p> <p>Parameters unknown to the forecaster: the reward distributions ν_1, \dots, ν_K of the arms.</p> <p>For each round $t = 1, 2, \dots, n$;</p> <p>(1) the forecaster chooses $I_t \in \{1, \dots, K\}$,</p> <p>(2) the environment draws the reward $X_{I_t, T_{I_t}(t)}$ from ν_{I_t} and independently of the past given I_t.</p> <p>At the end of the n rounds, the forecaster outputs a recommendation $J_n \in \{1, \dots, K\}$.</p>

Figure 1: The pure exploration problem for multi-armed bandits.

before the communication starts. The connection is then launched on the channel which is believed to be the best. Opportunistic communication systems rely on the same idea. Again the cumulative regret during the exploration phase is irrelevant since the user is only interested in the quality of its communication starting after the exploration phase.

More generally, the pure exploration problem addresses the design of strategies making the best possible use of available resources in order to optimize the performance of some decision-making task. That is, it occurs in situations with a preliminary exploration phase in which costs are not measured in terms of rewards but rather in terms of resources that come in limited budget (the number of patients in the test phase in the clinical trial setting and the time to connect in the communication example).

2 Problem setup

A stochastic multi-armed bandit game is parameterized by the number of arms K , the number of rounds (or budget) n , and K probability distributions ν_1, \dots, ν_K associated respectively with arm 1, \dots , arm K . These distributions are unknown to the forecaster. For $t = 1, \dots, n$, at round t , the forecaster chooses an arm I_t in the set of arms $\{1, \dots, K\}$, and observes a reward drawn from ν_{I_t} independently from the past (actions and observations). At the end of the n rounds, the forecaster selects an arm, denoted J_n , and is evaluated in terms of the difference between the mean reward of the optimal arm and the mean reward of J_n . Precisely, let μ_1, \dots, μ_K be the respective means of ν_1, \dots, ν_K . Let $\mu^* = \max_{k \in \{1, \dots, K\}} \mu_k$. The regret of the forecaster is

$$r_n = \mu^* - \mu_{J_n}.$$

For sake of simplicity, we will assume that the rewards are in $[0, 1]$ and that there is a unique optimal arm. Let i^* denote this arm (so, $\mu_{i^*} = \mu^*$). For $i \neq i^*$, we introduce the following suboptimality measure of arm i :

$$\Delta_i = \mu^* - \mu_i.$$

For reasons that will be obvious later, we also define Δ_{i^*} as the minimal gap

$$\Delta_{i^*} = \min_{i \neq i^*} \Delta_i.$$

We introduce the notation $(i) \in \{1, \dots, K\}$ to denote the i -th best arm (with ties break arbitrarily), hence

$$\Delta_{i^*} = \Delta_{(1)} = \Delta_{(2)} \leq \Delta_{(3)} \leq \dots \leq \Delta_{(K)}.$$

Let e_n denote the probability of error, that is the probability that the recommendation is not the optimal one:

$$e_n = \mathbb{P}(J_n \neq i^*).$$

We have $\mathbb{E}r_n = \sum_{i \neq i^*} \mathbb{P}(J_n = i) \Delta_i$, and consequently

$$\Delta_{i^*} e_n \leq \mathbb{E}r_n \leq e_n.$$

As a consequence of this equation, up to a second order term, e_n and $\mathbb{E}r_n$ behave similarly, and it does not harm to focus on the probability e_n .

For each arm i and all time rounds $t \geq 1$, we denote by $T_i(t)$ the number of times arm i was pulled from rounds 1 to t , and by $X_{i,1}, X_{i,2}, \dots, X_{i,T_i(t)}$ the sequence of associated rewards. Introduce $\hat{X}_{i,s} =$

$\frac{1}{s} \sum_{t=1}^s X_{i,t}$ the empirical mean of arm i after s pulls. In the following, the symbol c will denote a positive numerical constant which may differ from line to line.

The goal of this work is to propose allocation strategies with small regret, and possibly as small as the best allocation strategy which would know beforehand the distributions ν_1, \dots, ν_K up to a permutation. Before going further, note that the goal is unachievable for all distributions ν_1, \dots, ν_K : a policy cannot perform as well as the ‘‘oracle’’ allocation strategy in every particular cases. For instance, when the supports of ν_1, \dots, ν_K are disjoint, the oracle forecaster almost surely identifies an arm by a single draw of it. As a consequence, it has almost surely zero regret for any $n \geq K$. The generic policy which does not have any knowledge on the K distributions cannot reproduce this performance for any K -tuple of disjointly supported distributions. In this work, the above goal of deciding as well as an oracle will be reached for the set of Bernoulli distributions with parameters in $(0, 1)$, but the algorithms are defined for any distributions supported in $[0, 1]$.

We would like to mention that the case $K = 2$ is unique and simple since, as we will indirectly see, it is optimally solved by the uniform allocation strategy consisting in drawing each arm $n/2$ times (up to rounding problem), and at the end recommending the arm with the highest empirical mean. Therefore, our main contributions concern more the problem of the budget allocation when $K \geq 3$. The hardness of the task will be characterized by the following quantities

$$H_1 = \sum_{i=1}^K \frac{1}{\Delta_i^2} \quad \text{and} \quad H_2 = \max_{i \in \{1, \dots, K\}} i \Delta_{(i)}^{-2}.$$

These quantities are equivalent up to a logarithmic factor since we have (see Section 6.1)

$$H_2 \leq H_1 \leq \log(2K)H_2. \quad (1)$$

Intuitively, we will show that these quantities are indeed characteristic of the hardness of the problem, in the sense that they give the order of magnitude of the number of samples required to find the best arm with a reasonable probability. This statement will be made precise in the rest of the paper, in particular through Theorem 2 and Theorem 4.

Outline. In Section 3, we propose a highly exploring policy based on upper confidence bounds, called UCB-E (Upper Confidence Bound Exploration), in the spirit of UCB1 Auer et al. (2002). We prove that this algorithm, provided that it is appropriately tuned, has an upper bound on the probability of error e_n of order $\exp(-c \frac{n}{H_1})$. The core problem of this policy is the tuning of the parameter. The optimal value of the parameter depends on H_1 , which has no reason to be known beforehand by the forecaster, and which, to our knowledge, cannot be estimated from past observations with sufficiently high confidence in order that the resulting algorithm still satisfies a similar bound on e_n .

To get round this limitation, in Section 4, we propose a simple new policy called SR (Successive Rejects) that progressively rejects the arms which seem to be suboptimal. This algorithm is parameter-free and its probability of error e_n is at most of order $\exp(-\frac{n}{\log(2K)H_2})$. Since $H_2 \leq H_1 \leq \log(2K)H_2$, up to at most a logarithmic term in K , the algorithm performs as well as UCB-E while not requiring the knowledge of H_1 .

In Section 5, we prove that H_1 and H_2 truly represent the hardness of the problem (up to a logarithmic factor). Precisely, we consider a forecaster which knows the reward distributions of the arms *up to a permutation*. When these distributions are of Bernoulli type with parameter in $[p, 1-p]$ for some $p > 0$, there exists a permutation of the distributions for which the probability of error of the (oracle) forecaster is lower bounded by $\exp(-\frac{cn}{p(1-p)H_2})$.

Section 6 gathers some of the proofs. Section 7 provides some experiments testing the efficiency of the proposed policies and enlightening our theoretical results. We also discuss a modification of UCB-E where we perform a non-trivial online estimation of H_1 . We conclude in Section 8.

Example. To put in perspective the results we just mentioned, let us consider a specific example with Bernoulli distributions. Let $\nu_1 = \text{Ber}(\frac{1}{2})$, and $\nu_i = \text{Ber}(\frac{1}{2} - \frac{1}{K^i})$ for $i \in \{2, \dots, K\}$. Here, one can easily check that $H_2 = 2K^{2K}$. Thus, in this case, the probability of missing the best arm of SR is at most of order $\exp(-\frac{n}{2 \log(2K)K^{2K}})$. Moreover, in Section 5, we prove that there does not exist any forecaster (even with the knowledge of the distributions up to a permutation) with a probability of missing the best arm smaller than $\exp(-\frac{11n}{K^{2K}})$ for infinitely many n . Thus, our analysis finds that, for this particular reward distributions, the number of samples required to find the best arm is at least (of order of) K^{2K} , and SR actually finds it with (of order of) $\log(K)K^{2K}$ samples.

3 Highly exploring policy based on upper confidence bounds

In this section, we propose and study the algorithm UCB-E described in Figure 2. When a is taken of order $\log n$, the algorithm essentially corresponds to the UCB1 policy introduced in Auer et al. (2002), and

Parameter: exploration parameter $a > 0$.

For $i \in \{1, \dots, K\}$, let $B_{i,s} = \widehat{X}_{i,s} + \sqrt{\frac{a}{s}}$ for $s \geq 1$ and $B_{i,0} = +\infty$.

For each round $t = 1, 2, \dots, n$:
 Draw $I_t \in \operatorname{argmax}_{i \in \{1, \dots, K\}} B_{i, T_i(t-1)}$.

Let $J_n \in \operatorname{argmax}_{i \in \{1, \dots, K\}} \widehat{X}_{i, T_i(n)}$.

Figure 2: UCB-E (Upper Confidence Bound Exploration) algorithm.

its cumulative regret is of order $\log n$. Bubeck et al. (2009) have shown that algorithms having at most logarithmic cumulative regret, have at least a (non-cumulative) regret of order $n^{-\gamma}$ for some $\gamma > 0$. So taking a of order $\log n$ is inappropriate to reach exponentially small probability of error. For our regret notion, one has to explore much more and typically use a parameter which is essentially linear in n . Precisely, we have the following result, the proof of which can be found in Section 6.2.

Theorem 1 *If UCB-E is run with parameter $0 < a \leq \frac{25}{36} \frac{n-K}{H_1}$, then it satisfies*

$$e_n \leq 2nK \exp\left(-\frac{2a}{25}\right).$$

In particular for $a = \frac{25}{36} \frac{n-K}{H_1}$, we have $e_n \leq 2nK \exp\left(-\frac{n-K}{18H_1}\right)$.

The theorem shows that the probability of error of UCB-E is at most of order $\exp(-ca)$ for $a \geq \log n$. In fact, one can easily show a corresponding lower bound. In view of this, as long as $a \leq \frac{25}{36} \frac{n-K}{H_1}$, we can essentially say: the more we explore (i.e., the larger a is), the smaller the regret is. Besides, the smallest upper bound on the probability of error is obtained for a of order n/H_1 , and is therefore exponentially decreasing with n . The constant H_1 depends not only on how close the mean rewards of the two best arms are, but also on the number of arms and how close their mean reward is to the optimal mean reward. This constant should be seen as the order of the minimal number n for which the recommended arm is the optimal one with high probability. In Section 5, we will show that H_1 is indeed a good measure of the hardness of the task by showing that no forecaster satisfies $e_n \leq \exp\left(-\frac{cn}{H_2}\right)$ for any distributions ν_1, \dots, ν_K , where we recall that H_2 satisfies $H_2 \leq H_1 \leq \log(2K)H_2$.

One interesting message to take from the proof of Theorem 1 is that, with probability at least $1 - 2nK \exp\left(-\frac{2a}{25}\right)$, the number of draws of any suboptimal arm i is of order $a\Delta_i^{-2}$. This means that the optimal arm will be played at least $n - caH_1$, showing that for too small a , UCB-E “exploits” too much in view of our regret target. Theorem 1 does not specify how the algorithm performs when a is larger than $\frac{25}{36} \frac{n-K}{H_1}$. Nevertheless, similar arguments than the ones in the proof show that for large a , with high probability, only low rewarding arms are played of order $a\Delta_i^{-2}$ times, whereas the best ones are all drawn the same number of times up to a constant factor. The number of these similarly drawn arms grows with a . In the limit, when a goes to infinity, UCB-E is exactly the uniform allocation strategy studied in Bubeck et al. (2009). In general¹, the uniform allocation has a probability of error which can be lower and upper bounded by a quantity of the form $\exp\left(-c\frac{n\Delta_{i^*}^2}{K}\right)$. It consequently performs much worse than UCB-E for $a = \frac{25}{36} \frac{n-K}{H_1}$, since $H_1 \leq K\Delta_{i^*}^{-2}$, and potentially $H_1 \ll K\Delta_{i^*}^{-2}$ for very large number of arms with heterogeneous mean rewards.

One straightforward idea to cope with the absence of an oracle telling us the value of H_1 would be to estimate online the parameter H_1 and use this estimation in the algorithm. Unfortunately, we were not able to prove, and do not believe that, this modified algorithm generally attains the expected rate of convergence. Indeed, overestimating H_1 leads to low exploring, and in the event when the optimal arm has given abnormally low rewards, the arm stops being drawn by the policy, its estimated mean reward is thus not corrected, and the arm is finally not recommended by the policy. On the contrary, underestimating H_1 leads to draw too much the suboptimal arms, precluding a sufficiently accurate estimation of the mean rewards of the best arms. For this last case, things are in fact much more subtle than what can be retranscribed in these few lines,

¹We say “in general” to rule out some trivial cases (like when the reward distributions are all Dirac distributions) in which the probability of error e_n would be much smaller.

Let $A_1 = \{1, \dots, K\}$, $\overline{\log}(K) = \frac{1}{2} + \sum_{i=2}^K \frac{1}{i}$, $n_0 = 0$ and for $k \in \{1, \dots, K-1\}$,

$$n_k = \left\lceil \frac{1}{\overline{\log}(K)} \frac{n-K}{K+1-k} \right\rceil.$$

For each phase $k = 1, 2, \dots, K-1$:

- (1) For each $i \in A_k$, select arm i for $n_k - n_{k-1}$ rounds.
- (2) Let $A_{k+1} = A_k \setminus \arg \min_{i \in A_k} \widehat{X}_{i, n_k}$ (we only remove one element from A_k , if there is a tie, select randomly the arm to dismiss among the worst arms).

Let J_n be the unique element of A_K .

Figure 3: SR (Successive Rejects) algorithm.

and we notice that keeping track of a lower bound on H_1 would lead to the correct rate only under appropriate assumptions on the decrease of the sequence $\Delta_{(k)}$, $k \in \{1, \dots, K\}$. In Section 7 we push this idea and propose a way to estimate online H_1 , however we solely justify the corresponding algorithm by experiments. In the next section we propose an algorithm which does not suffer from these limitations.

4 Successive Rejects algorithm

In this section, we describe and analyze a new algorithm, SR (Successive Rejects), see Figure 3 for its precise description. Informally it proceeds as follows. First the algorithm divides the time (i.e., the n rounds) in $K-1$ phases. At the end of each phase, the algorithm dismisses the arm with the lowest empirical mean. During the next phase, it pulls equally often each arm which has not been dismissed yet. The recommended arm J_n is the last surviving arm. The length of the phases are carefully chosen to obtain an optimal (up to a logarithmic factor) convergence rate. More precisely, one arm is pulled $n_1 = \lceil \frac{1}{\overline{\log}(K)} \frac{n-K}{K} \rceil$ times, one $n_2 = \lceil \frac{1}{\overline{\log}(K)} \frac{n-K}{K-1} \rceil$ times, ..., and two arms are pulled $n_{K-1} = \lceil \frac{1}{\overline{\log}(K)} \frac{n-K}{2} \rceil$ times. SR does not exceed the budget of n pulls, since, from the definition $\overline{\log}(K) = \frac{1}{2} + \sum_{i=2}^K \frac{1}{i}$, we have

$$n_1 + \dots + n_{K-1} + n_{K-1} \leq K + \frac{n-K}{\overline{\log}(K)} \left(\frac{1}{2} + \sum_{k=1}^{K-1} \frac{1}{K+1-k} \right) = n.$$

For $K=2$, up to rounding effects, SR is just the uniform allocation strategy.

Theorem 2 *The probability of error of SR satisfies*

$$e_n \leq \frac{K(K-1)}{2} \exp\left(-\frac{n-K}{\overline{\log}(K)H_2}\right).$$

Proof: We can assume that the sequence of rewards for each arm is drawn before the beginning of the game. Thus the empirical reward for arm i after s pulls is well defined even if arm i has not been actually pulled s times. During phase k , at least one of the k worst arms is surviving. So, if the optimal arm i^* is dismissed at the end of phase k , it means that $\widehat{X}_{i^*, n_k} \leq \max_{i \in \{(K), (K-1), \dots, (K+1-k)\}} \widehat{X}_{i, n_k}$. By a union bound and Hoeffding's inequality, the probability of error $e_n = \mathbb{P}(A_K \neq \{i^*\})$ thus satisfies

$$\begin{aligned} e_n &\leq \sum_{k=1}^{K-1} \sum_{i=K+1-k}^K \mathbb{P}(\widehat{X}_{i^*, n_k} \leq \widehat{X}_{(i), n_k}) \\ &\leq \sum_{k=1}^{K-1} \sum_{i=K+1-k}^K \mathbb{P}(\widehat{X}_{(i), n_k} - \mu_{(i)} + \mu^* - \widehat{X}_{i^*, n_k} \geq \Delta_{(i)}) \\ &\leq \sum_{k=1}^{K-1} \sum_{i=K+1-k}^K \exp(-n_k \Delta_{(i)}^2) \leq \sum_{k=1}^{K-1} k \exp(-n_k \Delta_{(K+1-k)}^2). \end{aligned}$$

We conclude the proof by noting that by definition of n_k and H_2 , we have

$$n_k \Delta_{(K+1-k)}^2 \geq \frac{n-K}{\log(K)} \frac{1}{(K+1-k) \Delta_{(K+1-k)}^{-2}} \geq \frac{n-K}{\log(K) H_2}. \quad (2)$$

■

The following theorem provides a deeper understanding of how SR works. It lower bounds the sampling times of the arms and shows that at the end of phase k , we have a high-confidence estimation of $\Delta_{(K+1-k)}$ up to numerical constant factor. This intuition will prove to be useful in Section 7, see in particular Figure 4.

Theorem 3 *With probability at least $1 - \frac{K^3}{2} \exp\left(-\frac{n-K}{4\log(K)H_2}\right)$, for any arm j , we have*

$$T_j(n) \geq \frac{n-K}{4\log(K)H_2\Delta_j^2}. \quad (3)$$

With probability at least $1 - K^3 \exp\left(-\frac{n-K}{32\log(K)H_2}\right)$, for any $k \in \{1, \dots, K-1\}$, the dismissed arm $\ell_k = A_{k+1} \setminus A_k$ at the end of phase k satisfies

$$\frac{1}{4} \Delta_{(K+1-k)} \leq \frac{1}{2} \Delta_{\ell_k} \leq \max_{m \in A_k} \widehat{X}_{m, n_k} - \widehat{X}_{\ell_k, n_k} \leq \frac{3}{2} \Delta_{\ell_k} \leq 3 \Delta_{(K+1-k)}. \quad (4)$$

Proof: We consider the event \mathcal{E} on which for any $k \in \{1, \dots, K-1\}$, for any arm ℓ in the worst k arms, and any arm j such that $2\Delta_j \leq \Delta_\ell$, we have

$$\widehat{X}_{j, n_k} - \widehat{X}_{\ell, n_k} > 0.$$

This event holds with probability at least $1 - \frac{K^3}{2} \exp\left(-\frac{n-K}{4\log(K)H_2}\right)$, since, from Hoeffding's inequality, a union bound and (2), we have

$$\begin{aligned} & \sum_{k=1}^{K-1} \sum_{\ell \in \{(K), (K-1), \dots, (K+1-k)\}} \sum_{j: 2\Delta_j \leq \Delta_\ell} \mathbb{P}\left(\widehat{X}_{j, n_k} - \widehat{X}_{\ell, n_k} \leq 0\right) \\ & \leq \sum_{k=1}^{K-1} \sum_{\ell \in \{(K), (K-1), \dots, (K+1-k)\}} \sum_{j: 2\Delta_j \leq \Delta_\ell} \exp\left(-n_k(\Delta_\ell - \Delta_j)^2\right) \\ & \leq \sum_{k=1}^{K-1} kK \exp\left(-n_k \frac{\Delta_{(K+1-k)}^2}{4}\right) \leq \frac{K^3}{2} \exp\left(-\frac{n-K}{4\log(K)H_2}\right). \end{aligned}$$

During phase k , at least one of the k worst arms is surviving. On the event \mathcal{E} , this surviving arm has an empirical mean at the end of the phase which is smaller than the one of any arm j satisfying $2\Delta_j \leq \Delta_{(K+1-k)}$. So, at the end of phase k , any arm j satisfying $2\Delta_j \leq \Delta_{(K+1-k)}$ cannot be dismissed. Now, for a given arm j , we consider two cases depending whether there exists $m \in \{1, \dots, K\}$ such that $\Delta_{(m-1)} \leq 2\Delta_j \leq \Delta_{(m)}$. *First case.* If no such m exists, then we have $\Delta_j^2 T_j(n) \geq \frac{1}{4} \Delta_{(K)}^2 n_1 \geq \frac{n-K}{4\log(K)H_2}$, so that (3) holds.

Second case. If such m exists, then, from the above argument, the arm j cannot be dismissed before the end of the phase $K+2-m$ (since there exists $K+1-m$ arms ℓ such that $\Delta_\ell \geq 2\Delta_j$). From (2), we get

$$\Delta_j^2 T_j(n) \geq \Delta_j^2 n_{K+2-m} \geq \frac{\Delta_j^2}{\Delta_{(m-1)}^2} \frac{n-K}{\log(K)H_2} \geq \frac{n-K}{4\log(K)H_2},$$

which ends the proof of (3). We have seen that at the end of phase k , any arm j satisfying $2\Delta_j \leq \Delta_{(K+1-k)}$ cannot be dismissed. Consequently, at the end of phase k , the dismissed arm $\ell_k = A_{k+1} \setminus A_k$ satisfies the left inequality of

$$\frac{1}{2} \Delta_{(K+1-k)} \leq \Delta_{\ell_k} \leq 2\Delta_{(K+1-k)}. \quad (5)$$

Let us now prove the right inequality by contradiction. Consider k such that $2\Delta_{(K+1-k)} < \Delta_{\ell_k}$. Arm ℓ_k thus belongs to the $k-1$ worst arms. Hence, in the first $k-1$ rejects, say at the end of phase k' , an arm j with $\Delta_j \leq \Delta_{(K+1-k)}$ is dismissed. From the left inequality of (5), we get $\Delta_{(K+1-k')} \leq 2\Delta_j < \Delta_{\ell_k}$.

On the event \mathcal{E} , we thus have $\widehat{X}_{j,n_{k'}} - \widehat{X}_{\ell_k,n_{k'}} > 0$ (since ℓ_k belongs to the k' worst arms by the previous inequality). This contradicts the fact that j is rejected at phase k' . So (5) holds.

Now let \mathcal{E}' be the event on which for any arm j , and any $k \in \{1, \dots, K-1\}$ $|\widehat{X}_{j,n_k} - \mu_j| \leq \frac{\Delta_{(K+1-k)}}{8}$. Using again Hoeffding's inequality, a union bound and (2), this event holds with probability at least $1 - 2K(K-1) \exp\left(-\frac{n-K}{32\log(K)H_2}\right)$. We now work on the event $\mathcal{E} \cap \mathcal{E}'$, which holds with probability at least $1 - K^3 \exp\left(-\frac{n-K}{32\log(K)H_2}\right)$. From (5), the dismissed arm ℓ_k at the end of phase k satisfies

$$|\widehat{X}_{\ell_k,n_k} - \mu_{\ell_k}| \leq \frac{\Delta_{(K+1-k)}}{8} \leq \frac{\Delta_{\ell_k}}{4}.$$

Besides, we also have

$$\left| \max_{m \in A_k} \widehat{X}_{m,n_k} - \mu_{(1)} \right| \leq \frac{\Delta_{(K+1-k)}}{8} \leq \frac{\Delta_{\ell_k}}{4}.$$

Consequently, at the end of phase k , we have

$$\frac{1}{4} \Delta_{(K+1-k)} \leq \frac{1}{2} \Delta_{\ell_k} \leq \max_{m \in A_k} \widehat{X}_{m,n_k} - \widehat{X}_{\ell_k,n_k} \leq \frac{3}{2} \Delta_{\ell_k} \leq 3 \Delta_{(K+1-k)}.$$

■

5 Lower bound

In this section, we provide a general and somewhat surprising lower bound. We prove that, when the reward distributions are Bernoulli distributions with variances bounded away from 0, then for any forecaster, one can permute the distributions on the arms (before the game starts) so that the probability of missing the best arm will be at least of order $\exp\left(-\frac{cn}{H_2}\right)$. Note that, in this formulation, we allow the forecaster to *know* the reward distributions up to a permutation of the indices! However, as the lower bound expresses it, whatever Bernoulli distributions with variances bounded away from 0 are considered, the quantity H_2 is a good measure of the hardness of finding the best arm.

Theorem 4 (Lower Bound) *Let ν_1, \dots, ν_K be Bernoulli distributions with parameters in $[p, 1-p]$, $p \in (0, 1/2)$. For any forecaster, there exists a permutation $\sigma : \{1, \dots, K\} \rightarrow \{1, \dots, K\}$ such that the probability error of the forecaster on the bandit problem defined by $\tilde{\nu}_1 = \nu_{\sigma(1)}, \dots, \tilde{\nu}_K = \nu_{\sigma(K)}$ satisfies*

$$e_n \geq \exp\left(-\frac{(5+o(1))n}{p(1-p)H_2}\right),$$

where the $o(1)$ term depends only on K , p and n and goes to 0 when n goes to infinity (see the end of the proof).

The proof of this result is quite technical. However, it is simple to explain why we can expect such a bound to hold. Assume (without loss of generality) that the arms are ordered, i.e., $\mu_1 > \mu_2 \geq \dots \geq \mu_K$, and that all rewards $X_{i,t}$, $t \in \{1, \dots, n\}$, $i \in \{1, \dots, K\}$, are drawn before the game starts. Let $i \in \{2, \dots, K\}$. If $\widehat{X}_{1,n/i} < \widehat{X}_{i,n/i} \leq \widehat{X}_{j,n/i}$ for all $j \in \{2, \dots, i-1\}$, then it seems reasonable that a good forecaster should not pull arm 1 more than n/i times, and furthermore not select it as its recommendation. One can see that, the probability of the event we just described is of order of $\exp(-c(n/i)\Delta_i^2)$. Thus, with probability at least $\exp(-cn/\max_{2 \leq i \leq K} i\Delta_i^2)$, the forecaster makes an error, which is exactly the lower bound we propose. However, note that this argument does not yield a reasonable proof strategy, in particular we assumed a “good” forecaster with a “reasonable” behavior. For instance, it is obvious that the proof has to permute the arms, since a forecaster could, despite all, choose arm 1 as its recommendation, which imply a probability error of 0 as soon as the best arm is in position 1.

The main idea of our proposed proof goes as follows. A bandit problem is defined by a product distribution $\nu = \nu_1 \otimes \dots \otimes \nu_K$. One can consider that at the beginning of the game, n K -tuples of rewards are sampled from this product distribution. This defines a table of nK rewards. A forecaster will explore a subpart of this table. We want to find a permutation σ of $\{1, \dots, K\}$ so that the indices of the best arm for ν and $\tilde{\nu} = \nu_{\sigma(1)} \otimes \dots \otimes \nu_{\sigma(K)}$ are different and such that the likelihood ratio of the explored part of the table of nK rewards under ν and $\tilde{\nu}$ is at least of order $\exp(-cn/H_2)$ with probability with respect to $\nu^{\otimes n}$ lower bounded by a positive numerical constant. This would imply the claimed bound. Remark that, the “likelihood cost” of moving distribution ν_i to arm j depends on both the (Kullback-Leibler) distance between the distributions ν_i and ν_j , and the number of times arm j is pulled. Thus, we have to find the right trade-off between moving

a distribution to a “close” distribution, and the fact that the target arm should not be pulled too much. To do this, we “slice” the set of indices in a non-trivial (and non-intuitive) way. This “slicing” depends only on the reward distributions, and not on the considered forecaster. Then, to put it simply, we move the less drawn arm from one slice to the less drawn arm in the next slice. Note that the preceding sentence is not well defined, since by doing this we would get a random permutation (which of course does not make sense to derive a lower bound). However, at the cost of some technical difficulties, it is possible to circumvent this issue.

To achieve the program outlined above, as already hinted, we use the Kullback-Leibler divergence, which is defined for two probability distributions ρ, ρ' on $[0, 1]$ with ρ absolutely continuous with respect to ρ' as:

$$\text{KL}(\rho, \rho') = \int_0^1 \log \left(\frac{d\rho}{d\rho'}(x) \right) d\rho(x) = \mathbb{E}_{X \sim \rho} \log \left(\frac{d\rho}{d\rho'}(X) \right).$$

Another quantity of particular interest for our analysis is $\widehat{\text{KL}}_{i,t}(\rho, \rho') = \sum_{s=1}^t \log \left(\frac{d\rho}{d\rho'}(X_{i,s}) \right)$. In particular, note that, if arm i has distribution ρ , then this quantity represents the (non re-normalized) empirical estimation of $\text{KL}(\rho, \rho')$ after t pulls of arm i . Let \mathbb{P}_ν and \mathbb{E}_ν the probability and expectation signs when we integrate with respect to the distribution $\nu^{\otimes n}$. Another important property is that for any two product distributions ν, ν' , which differ only on index i , and for any event A , one has:

$$\mathbb{P}_\nu(A) = \mathbb{E}_{\nu'} \mathbf{1}_A \exp \left(-\widehat{\text{KL}}_{i,T_i(n)}(\nu'_i, \nu_i) \right), \quad (6)$$

since we have $\prod_{s=1}^{T_i(n)} \frac{d\nu_i}{d\nu'_i}(X_{i,s}) = \exp \left(-\widehat{\text{KL}}_{i,T_i(n)}(\nu'_i, \nu_i) \right)$.

Proof: First step: Notations. Without loss of generality we can assume that ν is ordered in the sense that $\mu_1 > \mu_2 \geq \dots \geq \mu_K$. Moreover let $L \in \{2, \dots, K\}$ such that $H_2 = L/\Delta_L^2$, that is for all $i \in \{1, \dots, K\}$,

$$i/\Delta_i^2 \leq L/\Delta_L^2. \quad (7)$$

We define now recursively the following sets. Let $k_1 = 1$,

$$\Sigma_1 = \left\{ i : \mu_L \leq \mu_i \leq \mu_L + \frac{\Delta_L}{L^{1/2^{k_1}}} \right\},$$

and for $j > 1$,

$$\Sigma_j = \left\{ i : \mu_L + \frac{\Delta_L}{L^{1/2^{k_{j-1}}}} < \mu_i \leq \mu_L + \frac{\Delta_L}{L^{1/2^{k_j}}} \right\},$$

where k_j is the smallest integer (if it exists, otherwise set $k_j = +\infty$) such that $|\Sigma_j| > 2|\Sigma_{j-1}|$. Let $\ell = \max\{j : k_j < +\infty\}$. We define now the random variables Z_1, \dots, Z_ℓ corresponding to the indices of the less sampled arms of the respective slices $\Sigma_1, \dots, \Sigma_\ell$: for $j \in \{1, \dots, \ell\}$,

$$Z_j \in \underset{i \in \Sigma_j}{\text{argmin}} T_i(n).$$

Finally let $Z_{\ell+1} \in \underset{i \in \{1, \dots, L\} \setminus \{J_n\}}{\text{argmin}} T_i(n)$.

Second step: Controlling $T_{Z_j}(n)$, $j \in \{1, \dots, \ell+1\}$. We first prove that for any $j \in \{1, \dots, \ell\}$,

$$3|\Sigma_j| \geq L^{1 - \frac{1}{2^{k_{j+1}-1}}}. \quad (8)$$

To do so let us note that, by definition of k_{j+1} , we have

$$\begin{aligned} 2|\Sigma_j| &\geq \left| \left\{ i : \mu_L + \Delta_L/L^{1/2^{k_j}} < \mu_i \leq \mu_L + \Delta_L/L^{1/2^{k_{j+1}-1}} \right\} \right| \\ &\geq \left| \left\{ i : \mu_i \leq \mu_L + \Delta_L/L^{1/2^{k_{j+1}-1}} \right\} \right| - (|\Sigma_1| + \dots + |\Sigma_{j-1}|). \end{aligned}$$

Now remark that, by definition again, we have $|\Sigma_1| + \dots + |\Sigma_{j-1}| \leq (2^{-(j-1)} + \dots + 2^{-1})|\Sigma_j| \leq |\Sigma_j|$. Thus we obtain $3|\Sigma_j| \geq \left| \left\{ i : \mu_i \leq \mu_L + \Delta_L/L^{1/2^{k_{j+1}-1}} \right\} \right|$. We finish the proof of (8) with the following calculation, which makes use of (7). For any $v \geq 1$,

$$\begin{aligned} |\{i : \mu_i \leq \mu_L + \Delta_L/v\}| &= |\{i : \Delta_i \geq \Delta_L(1 - 1/v)\}| \\ &\geq \left| \left\{ i : \sqrt{\frac{i}{L}} \Delta_L \geq \Delta_L(1 - 1/v) \right\} \right| \\ &= |\{i : i \geq L(1 - 1/v)^2\}| \geq L \left(1 - (1 - 1/v)^2 \right) \geq L/v. \end{aligned}$$

Now (8) directly entails (since a minimum is smaller than an average), for $j \in \{1, \dots, \ell\}$,

$$T_{Z_j}(n) \leq 3L \frac{1}{2^{k_{j+1}-1}} \sum_{i \in \Sigma_j} T_i(n). \quad (9)$$

Besides, since $Z_{\ell+1}$ is the less drawn arm among $L-1$ arms, we trivially have

$$T_{Z_{\ell+1}}(n) \leq \frac{n}{L-1}. \quad (10)$$

Third step: A change of measure. Let $\nu' = \nu_L \otimes \nu_2 \otimes \dots \otimes \nu_K$ be a modified product distribution where we replaced the best distribution by ν_L . Now let us consider the event

$$C_n = \left\{ \forall t \in \{1, \dots, n\}, i \in \{2, \dots, L\}, j \in \{1, \dots, L\}, \right. \\ \left. \widehat{\text{KL}}_{i,t}(\nu_i, \nu_j) \leq t \text{KL}(\nu_i, \nu_j) + o_n \quad \text{and} \quad \widehat{\text{KL}}_{1,t}(\nu_L, \nu_j) \leq t \text{KL}(\nu_L, \nu_j) + o_n \right\},$$

where $o_n = 2 \log(p^{-1}) \sqrt{n \log(2L)}$. From Hoeffding's maximal inequality, one can prove that we have $\mathbb{P}_{\nu'}(C_n) \geq 1/2$. We thus have $\sum_{1 \leq z_1, \dots, z_{\ell+1} \leq L} \mathbb{P}_{\nu'}(C_n \cap \{Z_1 = z_1, \dots, Z_{\ell+1} = z_{\ell+1}\}) \geq 1/2$. Moreover note that Z_1, \dots, Z_{ℓ} are all distinct. Thus there exist $\ell+1$ constants $z_1, \dots, z_{\ell+1}$ such that, for $A_n = C_n \cap \{Z_1 = z_1, \dots, Z_{\ell+1} = z_{\ell+1}\}$, we have

$$\mathbb{P}_{\nu'}(A_n) \geq \frac{1}{2L \times L!}. \quad (11)$$

Since, by definition $Z_{\ell+1} \neq J_n$, we have

$$A_n \subset \{J_n \neq z_{\ell+1}\}. \quad (12)$$

In the following we treat differently the cases $z_{\ell+1} = 1$ and $z_{\ell+1} \neq 1$. First, let us assume that $z_{\ell+1} = 1$. Then, an application of (6) and (12) directly gives, by definition of A_n ,

$$\begin{aligned} e_n(\nu) = \mathbb{P}_{\nu}(J_n \neq 1) &= \mathbb{E}_{\nu'} \mathbb{1}_{J_n \neq 1} \exp\left(-\widehat{\text{KL}}_{1, T_1(n)}(\nu_L, \nu_1)\right) \\ &\geq \mathbb{E}_{\nu'} \mathbb{1}_{A_n} \exp\left(-\widehat{\text{KL}}_{1, T_1(n)}(\nu_L, \nu_1)\right) \\ &\geq \mathbb{E}_{\nu'} \mathbb{1}_{A_n} \exp\left(-o_n - T_{Z_{\ell+1}}(n) \text{KL}(\nu_L, \nu_1)\right) \\ &\geq \frac{1}{2L \times L!} \exp\left(-o_n - \frac{n}{L-1} \text{KL}(\nu_L, \nu_1)\right), \end{aligned}$$

where we used (10) and (11) for the last equation. Now, for any $p, q \in [0, 1]$, the KL divergence between Bernoulli distributions of parameters p and q satisfies

$$\text{KL}(\text{Ber}(p), \text{Ber}(q)) \leq \frac{(p-q)^2}{q(1-q)}. \quad (13)$$

This can be seen by using $\log u \leq u-1$ on the two logarithmic terms in $\text{KL}(\text{Ber}(p), \text{Ber}(q))$. In particular, it implies $\text{KL}(\nu_L, \nu_1) \leq \frac{\Delta_L^2}{p(1-p)}$, which concludes the proof in the case $z_{\ell+1} = 1$.

Assume now that $z_{\ell+1} \neq 1$. In this case we prove that the lower bound holds for a well defined permuted product distribution $\tilde{\nu}$ of ν . We define it as follows. Let m be the smallest $j \in \{1, \dots, \ell+1\}$ such that $z_m = z_{\ell+1}$. Now we set $\tilde{\nu}$ as follows: $\tilde{\nu}_{z_m} = \nu_1, \tilde{\nu}_{z_{m-1}} = \nu_{z_m}, \dots, \tilde{\nu}_{z_1} = \nu_{z_2}, \tilde{\nu}_1 = \nu_{z_1}$, and $\tilde{\nu}_j = \nu_j$ for other values of j in $\{1, \dots, K\}$. Remark that $\tilde{\nu}$ is indeed the result of a permutation of the distributions of ν . Again, an application of (6) and (12) gives, by definition of A_n ,

$$\begin{aligned} e_n(\tilde{\nu}) = \mathbb{P}_{\tilde{\nu}}(J_n \neq z_m) &= \mathbb{E}_{\nu'} \mathbb{1}_{J_n \neq z_m} \exp\left(-\widehat{\text{KL}}_{1, T_1(n)}(\nu_L, \nu_{z_1}) - \sum_{j=1}^{m-1} \widehat{\text{KL}}_{z_j, T_{z_j}(n)}(\nu_{z_j}, \nu_{z_{j+1}}) - \widehat{\text{KL}}_{z_m, T_{z_m}(n)}(\nu_{z_m}, \nu_{z_1})\right) \\ &\geq \mathbb{E}_{\nu'} \mathbb{1}_{A_n} \exp\left(- (m+1)o_n - T_1(n) \text{KL}(\nu_L, \nu_{z_1}) - \sum_{j=1}^{m-1} T_{Z_j}(n) \text{KL}(\nu_{Z_j}, \nu_{Z_{j+1}}) \right. \\ &\quad \left. - T_{Z_m}(n) \text{KL}(\nu_{Z_m}, \nu_{Z_1})\right). \quad (14) \end{aligned}$$

From (13), the definition of Σ_j , and since the parameters of the Bernoulli distributions are in $[p, 1 - p]$, we have $\text{KL}(\nu_L, \nu_{Z_1}) \leq \frac{1}{p(1-p)} \frac{\Delta_L^2}{L}$, $\text{KL}(\nu_{Z_m}, \nu_{Z_1}) \leq \frac{\Delta_L^2}{p(1-p)}$, and for any $j \in \{1, \dots, m-1\}$,

$$\text{KL}(\nu_{Z_j}, \nu_{Z_{j+1}}) \leq \frac{1}{p(1-p)} \left(\frac{\Delta_L}{L^{1/2^{k_{j+1}}}} \right)^2.$$

Reporting these inequalities, as well as (9), (10) and (11) in (14), we obtain:

$$\begin{aligned} e_n(\tilde{\nu}) &\geq \mathbb{E}_{\nu'} \mathbb{1}_{A_n} \exp \left(-(m+1)o_n - 3 \frac{\Delta_L^2}{p(1-p)L} \left(T_1(n) + \sum_{j=1}^{m-1} \sum_{i \in \Sigma_j} T_i(n) + \frac{nL}{3(L-1)} \right) \right) \\ &\geq \frac{1}{2L \times L!} \exp \left(-L o_n - 3n \frac{\Delta_L^2}{p(1-p)L} \left(1 + \frac{L}{3(L-1)} \right) \right) \end{aligned}$$

Since $L \leq K$ and $2K \times K! \leq \exp(2K \log(K))$ and from the definitions of o_n and L , we obtain

$$e_n(\tilde{\nu}) \geq \exp \left(-2K \log(K) - 2K \log(p^{-1}) \sqrt{n \log(2K)} - 5 \frac{n}{p(1-p)H_2} \right),$$

which concludes the proof. ■

6 Proofs

6.1 Proof of Inequalities (1)

Let $\overline{\log}(K) = \frac{1}{2} + \sum_{i=2}^K \frac{1}{i}$. Remark that $\log(K+1) - 1/2 \leq \overline{\log}(K) \leq \log(K) + 1/2 \leq \log(2K)$. Precisely, we will prove

$$H_2 \leq H_1 \leq \overline{\log}(K) H_2,$$

which is tight to the extent that the right inequality is an equality when for some $0 < c \leq 1/\sqrt{K}$, we have $\Delta_{(i)} = \sqrt{ic}$ for any $i \neq i^*$, and the left inequality is an equality if all Δ_i 's are equal.

Proof: The left inequality follows from: for any $i \in \{1, \dots, K\}$, $H_1 = \sum_{k=1}^K \Delta_{(k)}^{-2} \geq \sum_{k=1}^i \Delta_{(k)}^{-2} \geq i \Delta_{(i)}^{-2}$. The right inequality directly comes from $\sum_{i=1}^K \Delta_{(i)}^{-2} = \Delta_{(2)}^{-2} + \sum_{i=2}^K \frac{1}{i} i \Delta_{(i)}^{-2} \leq \overline{\log}(K) \max_{i \in \{1, \dots, K\}} i \Delta_{(i)}^{-2}$.

6.2 Proof of Theorem 1

First step. Let us consider the event

$$\xi = \left\{ \forall i \in \{1, \dots, K\}, s \in \{1, \dots, n\}, |\widehat{X}_{i,s} - \mu_i| < \frac{1}{5} \sqrt{\frac{a}{s}} \right\}.$$

From Hoeffding's inequality and a union bound, we have $\mathbb{P}(\xi) \geq 1 - 2nK \exp\left(-\frac{2a}{25}\right)$. In the following, we prove that on the event ξ we have $J_n = i^*$, which concludes the proof. Since J_n is the empirical best arm, and given that we are on ξ , it is enough to prove that

$$\frac{1}{5} \sqrt{\frac{a}{T_i(n)}} \leq \frac{\Delta_i}{2}, \forall i \in \{1, \dots, K\},$$

or equivalently:

$$T_i(n) \geq \frac{4}{25} \frac{a}{\Delta_i^2}, \forall i \in \{1, \dots, K\}. \quad (15)$$

Second step. Firstly we prove by induction that

$$T_i(t) \leq \frac{36}{25} \frac{a}{\Delta_i^2} + 1, \forall i \neq i^*. \quad (16)$$

It is obviously true at time $t = 1$. Now assume that the formula is true at time $t - 1$. If $I_t \neq i$ then $T_i(t) = T_i(t - 1)$ and the formula still holds. On the other hand, if $I_t = i$, then in particular it means that $B_{i, T_i(t-1)} \geq B_{i^*, T_{i^*}(t-1)}$. Moreover, since we are on ξ , we have $B_{i^*, T_{i^*}(t-1)} \geq \mu^*$ and $B_{i, T_i(t-1)} \leq \mu_i + \frac{6}{5} \sqrt{\frac{a}{T_i(t-1)}}$. Thus, we have $\frac{6}{5} \sqrt{\frac{a}{T_i(t-1)}} \geq \Delta_i$. By using $T_i(t) = T_i(t - 1) + 1$, we obtain (16).

Parameter: exploration rate $c > 0$.

Definitions: For $k \in \{1, \dots, K-1\}$, let $n_k = \lceil \frac{1}{\log(K)} \frac{n-K}{K+1-k} \rceil$, $t_0 = 0$, $t_1 = Kn_1$, and for $k > 1$, $t_k = n_1 + \dots + n_{k-1} + (K-k+1)n_k$. For $i \in \{1, \dots, K\}$ and $a > 0$, let $B_{i,s}(a) = \widehat{X}_{i,s} + \sqrt{\frac{a}{s}}$ for $s \geq 1$ and $B_{i,0} = +\infty$.

Algorithm: For each phase $k = 0, 1, \dots, K-1$:
Let $\widehat{H}_{1,k} = K$ if $k = 0$, and otherwise

$$\widehat{H}_{1,k} = \max_{K-k+1 \leq i \leq K} i \widehat{\Delta}_{\langle i \rangle}^{-2},$$

where $\widehat{\Delta}_i = (\max_{1 \leq j \leq K} \widehat{X}_{j, T_j(t_k)}) - \widehat{X}_{i, T_i(t_k)}$ and $\langle i \rangle$ is an ordering such that $\widehat{\Delta}_{\langle 1 \rangle} \leq \dots \leq \widehat{\Delta}_{\langle K \rangle}$.

For $t = t_k + 1, \dots, t_{k+1}$:

Draw $I_t \in \operatorname{argmax}_{i \in \{1, \dots, K\}} B_{i, T_i(t-1)}(cn/\widehat{H}_{1,k})$.

Recommendation: Let $J_n \in \operatorname{argmax}_{i \in \{1, \dots, K\}} \widehat{X}_{i, T_i(n)}$.

Figure 4: Adaptive UCB-E algorithm. Its intuitive justification goes as follows: The time points t_k correspond to the moments where the Successive Rejects algorithm would dismiss an arm. Intuitively, in light of Theorem 3, one can say that at time t_k a good algorithm should have reasonable approximation of the gaps between the best arm and the k worst arms, that is the quantities $\Delta_{(K-k+1)}, \dots, \Delta_{(K)}$. Now with these quantities, one can build a lower estimate of H_2 and thus also of H_1 . We use this estimate between the time points t_k and t_{k+1} to tune the parameter a of UCB-E.

Now we prove an other useful formula:

$$T_i(t) \geq \frac{4}{25} \min \left(\frac{a}{\Delta_i^2}, \frac{25}{36} (T_{i^*}(t) - 1) \right), \forall i \neq i^*. \quad (17)$$

With the same inductive argument as the one to get equation (16), we only need to prove that this formula holds when $I_t = i^*$. By definition of the algorithm, and since we are on ξ , when $I_t = i^*$ we have for all i :

$$\mu^* + \frac{6}{5} \sqrt{\frac{a}{T_{i^*}(t-1)}} \geq \mu_i + \frac{4}{5} \sqrt{\frac{a}{T_i(t-1)}},$$

which implies

$$T_i(t-1) \geq \frac{16}{25} \frac{a}{\left(\Delta_i + \frac{6}{5} \sqrt{\frac{a}{T_{i^*}(t-1)}} \right)^2}.$$

We then obtain (17) by using $u + v \leq 2 \max(u, v)$, $T_i(t) = T_i(t-1)$ and $T_{i^*}(t-1) = T_{i^*}(t) - 1$.

Third step. Recall that we want to prove equation (15). From (17), we only have to show that

$$\frac{25}{36} (T_{i^*}(n) - 1) \geq \frac{a}{\Delta_{i^*}^2},$$

where we recall that Δ_{i^*} is the minimal gap $\Delta_{i^*} = \min_{i \neq i^*} \Delta_i$. Using equation (16) we obtain:

$$T_{i^*}(n) - 1 = n - 1 - \sum_{i \neq i^*} T_i(n) \geq n - K - \frac{36}{25} a \sum_{i \neq i^*} \Delta_i^{-2} \geq \frac{36}{25} a \Delta_{i^*}^{-2},$$

where the last inequality uses $\frac{36}{25} H_1 a \leq n - K$. This concludes the proof.

7 Experiments

We propose a few simple experiments to illustrate our theoretical analysis. As a baseline comparison we use the Hoeffding Race algorithm, see Maron and Moore (1993), and the uniform strategy, which pulls equally often each arm and recommend the arm with the highest empirical mean, see Bubeck et al. (2009) for its theoretical analysis. We consider only Bernoulli distributions, and the optimal arm always has parameter $1/2$. Each experiment corresponds to a different situation for the gaps, they are either clustered in few groups,

or distributed according to an arithmetic or geometric progression. In each experiment we choose the number of samples (almost) equal to H_1 (except for the last experiment where we run it twice, the second time with $2H_1$ samples). If our understanding of the meaning of H_1 is sound, in each experiment the strategies SR and UCB-E should be able to find the best arm with a reasonable probability (which should be roughly of the same order in each experiment). We report our results in Figure 5. The parameters for the experiments are as follows:

- Experiment 1: One group of bad arms, $K = 20$, $\mu_{2:20} = 0.4$ (meaning for any $j \in \{2, \dots, 20\}$, $\mu_j = 0.4$)
- Experiment 2: Two groups of bad arms, $K = 20$, $\mu_{2:6} = 0.42$, $\mu_{7:20} = 0.38$.
- Experiment 3: Geometric progression, $K = 4$, $\mu_i = 0.5 - (0.37)^i$, $i \in \{2, 3, 4\}$.
- Experiment 4: 6 arms divided in three groups, $K = 6$, $\mu_2 = 0.42$, $\mu_{3:4} = 0.4$, $\mu_{5:6} = 0.35$.
- Experiment 5: Arithmetic progression, $K = 15$, $\mu_i = 0.5 - 0.025i$, $i \in \{2, \dots, 15\}$.
- Experiment 6: Two good arms and a large group of bad arms, $K = 20$, $\mu_2 = 0.48$, $\mu_{3:20} = 0.37$.
- Experiment 7: Three groups of bad arms, $K = 30$, $\mu_{2:6} = 0.45$, $\mu_{7:20} = 0.43$, $\mu_{21:30} = 0.38$.

The different graphics should be read as follows: Each bar represents a different algorithm and the bar's height represents the probability of error of this algorithm. The correspondence between algorithms and bars is the following:

- Bar 1: Uniform sampling strategy.
- Bar 2-4: Hoeffding Race algorithm with parameters $\delta = 0.01, 0.1, 0.3$.
- Bar 5: Successive Rejects strategy.
- Bar 6-9: UCB-E with parameter $a = cn/H_1$ where respectively $c = 1, 2, 4, 8$.
- Bar 10-14: Adaptive UCB-E (see Figure 4) with parameters $c = 1/4, 1/2, 1, 2, 4$.

8 Conclusion

This work has investigated strategies for finding the best arm in a multi-armed bandit problem. It has proposed a simple parameter-free algorithm, SR, that attains optimal guarantees up to a logarithmic term (Theorem 2 and Theorem 4). A precise understanding of both SR (Theorem 3) and a UCB policy (Theorem 1) lead us to define a new algorithm, Adaptive UCB-E. It comes without guarantee of optimal rates (see end of Section 3), but performs better than SR in practice (for $c = 1$, Adaptive UCB-E outperformed SR on all the experiments we did, even those done to make it fail). One possible explanation is that SR is too static: it does not implement more data driven arguments such as: in a phase, a surviving arm performing much worse than the other ones is still drawn until the end of the phase even if it is clear that it is the next dismissed arm.

Extensions of this work may concentrate on the following problems. (i) What is a good measure of hardness when one takes into account the (empirical) variances? Do we have a good scaling with respect to the variance with the current algorithms or do we need to modify them? (ii) Is it possible to derive a natural anytime version of Successive Rejects (without using a doubling trick)? (iii) Is it possible to close the logarithmic gap between the lower and upper bounds? (iv) How should we modify the algorithm and the analysis if one is interested in recommending the top m actions instead of a single one?

References

- P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning Journal*, 47(2-3):235–256, 2002.
- S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *Proc. of the 20th International Conference on Algorithmic Learning Theory*, 2009.
- C. Domingo, R. Gavaldà, and O. Watanabe. Adaptive sampling methods for scaling up knowledge discovery algorithms. *Data Mining and Knowledge Discovery*, 6(2):131–152, 2002.
- E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *The Journal of Machine Learning Research*, 7:1079–1105, 2006.
- O. Maron and A. W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *NIPS*, pages 59–66, 1993.
- V. Mnih, Cs. Szepesvári, and J.-Y. Audibert. Empirical bernstein stopping. In *ICML*, volume 307, pages 672–679, 2008.
- H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematics Society*, 58:527–535, 1952.

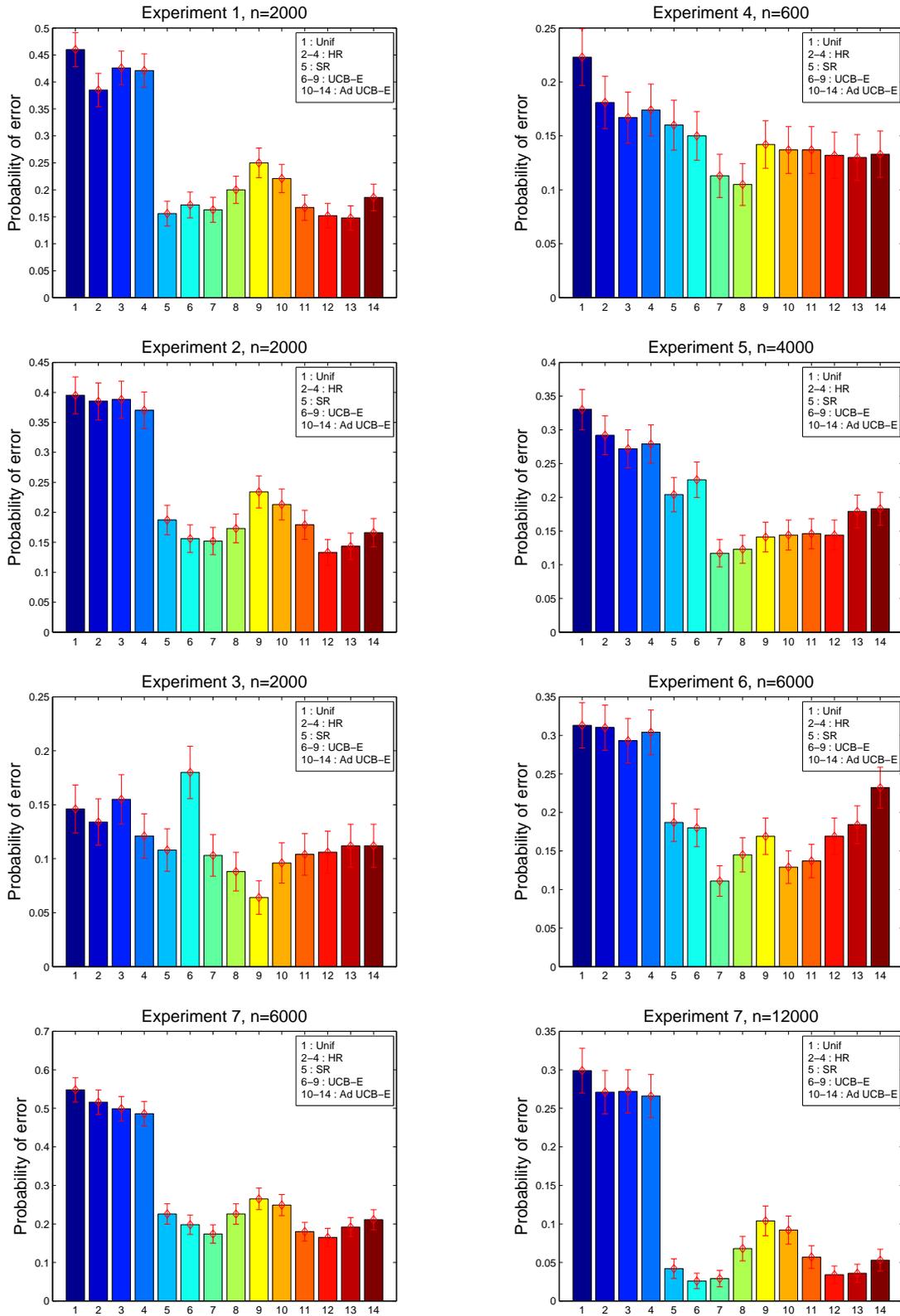


Figure 5: These results support our theoretical findings in the following sense: Despite the fact that the experiments are very different, one can see that since we use a number of samples (almost) equal to the hardness H_1 , in all of them we get a probability of error of the same order, and moreover this probability is small enough to say that we identified the best arm. Note that the Successive Rejects algorithm represents in all cases a substantial improvement over both the naive uniform strategy and Hoeffding Race. These results also justify experimentally the algorithm Adaptive UCB-E. Indeed one can see that with the constant $c = 1$, we obtain better results than SR in all experiments, even in experiment 6 which was designed to be a difficult instance of Adaptive UCB-E.

Nonparametric Bandits with Covariates

Philippe Rigollet*

Operations Research & Financial Engineering
Princeton University
Princeton, NJ 08544
rigollet@princeton.edu

Assaf Zeevi

Graduate School of Business
Columbia University
New York, NY 10027
assaf@gsb.columbia.edu

Abstract

We consider a bandit problem which involves sequential sampling from two populations (arms). Each arm produces a noisy reward realization which depends on an observable random *covariate*. The goal is to maximize cumulative expected reward. We derive general lower bounds on the performance of any admissible policy, and develop an algorithm whose performance achieves the order of said lower bound up to logarithmic terms. This is done by decomposing the global problem into suitably “localized” bandit problems. Proofs blend ideas from nonparametric statistics and traditional methods used in the bandit literature.

1 Introduction

The seminal paper of Robbins (1952) introduced an important class of sequential optimization problems, otherwise known as multi-armed bandits. These models have since been used extensively in such fields as statistics, operations research, engineering, computer science and economics. The traditional two-armed bandit problem can be described as follows. Consider two statistical populations (arms), where at each point in time it is possible to sample from only one of the two and receive a random reward dictated by the properties of the sampled population. The objective is to devise a sampling policy that maximizes expected cumulative (or discounted) rewards over a finite (or infinite) time horizon. The difference between the performance of said sampling policy and that of an oracle, that repeatedly samples from the population with the higher mean reward, is called the *regret*. Thus, one can re-phrase the objective as minimizing the regret.

The original motivation for bandit-type problems originates from treatment allocation in clinical trials; see, e.g., Lai and Robbins (1985) for further discussion and references therein. Here patients enter sequentially and receive one of several treatments. The efficacy of each treatment is unknown, and for each patient a noisy measurement of it is recorded. The goal is to assign as many patients as possible to the best treatment. An example of more recent work can be found in the area of web-based advertising, and more generally customized marketing. An on-line publisher needs to choose one of several ads to present to consumers, where the efficacy of these ads is unknown. The publisher observes click-through-rates (CTRs) for each ad, which provide a noisy measurement of the efficacy, and based on that needs to assign ads that maximize CTR.

When the populations being sampled are homogenous, i.e., when the sequential rewards are independent and identically distributed (iid) in each arm, Lai and Robbins (1985) proposed a family of policies that at each step compute the empirical mean reward in each arm, and adds to that a confidence bound that accounts for uncertainty in these estimates. These so-called upper-confidence-bound (UCB) policies were shown to be asymptotically optimal. In particular, it is proven in Lai and Robbins (1985) that such a policy incurs a regret of order $\log n$, where n is the length of the time horizon, and no other “good” policy can (asymptotically) achieve a smaller regret; see also Auer et al. (2002). The elegance of the theory and sharp results developed in Lai and Robbins (1985) hinge to a large extent on the assumption of homogenous populations and hence identically distributed rewards. This, however, is clearly too restrictive for many applications of interest. Often, the decision maker observes further information and based on that a more *customized* allocation can be made. In such settings rewards may still be assumed to be independent, but no longer identically distributed in each arm. A particular way to encode this is to allow for an exogenous variable (a covariate) that affects the rewards generated by each arm at each point in time when this arm is pulled.

Such a formulation was first introduced in Woodroffe (1979) under parametric assumptions and in a somewhat restricted setting; see Goldenshluger and Zeevi (2009) and Wang et al. (2005) for two very different

*Partially supported by NSF (DMS-0906424).

recent approaches to the study of such bandit problems, as well as references therein for further links to antecedent literature. The first work to venture outside the realm of parametric modeling assumptions was that of Yang and Zhu (2002). In particular, they assumed the mean response in each arm, conditional on the covariate value, follows a general functional form, hence one can view their setting as a *nonparametric* bandit problem. They proposed a policy that is based on estimating each response function, and then, rather than greedily choosing the arm with the highest estimated mean response given the covariate, allows with some small probability of selecting a potentially inferior arm. (This is a variant of ϵ -greedy policies; see Auer et al. (2002).) If the nonparametric estimators of the arms' functional response are consistent, and the randomization is chosen in a suitable manner, then the above policies ensure that the average regret tends to zero as the time horizon n grows to infinity. In the typical bandit terminology, such policies are said to be *consistent*. However, it is unclear whether they satisfy a more refined notion of optimality, insofar as the magnitude of the regret is concerned, as is the case for UCB-type policies in traditional bandit problems. Moreover, the study by Yang and Zhu (2002) does not spell out the connection between the characteristics of the class of response functions, and the resulting complexity of the nonparametric bandit problem.

The purpose of the present paper is to further understanding of nonparametric bandit problems, deriving regret-optimal policies and shedding light on some of the elements that dictate the complexity of such problems. We make only two assumptions on the underlying functional form that governs the arms' responses. The first is a mild smoothness condition. Smoothness assumptions can be exploited using "plug-in" policies as opposed "minimum contrast" policies; a detailed account of the differences and similarities between these two setups in the full information case can be found in Audibert and Tsybakov (2007). Minimum contrast type policies have already received some attention in the bandit literature with side information, aka *contextual bandits*, in the papers of Langford and Zhang (2008) and also Kakade et al. (2008). In these studies, admissible policies are restricted to a more limited set than the general class of non-anticipating policies. A related problem online convex optimization with side information was studied by Hazan and Megiddo (2007), where the authors use discretization technique similar to the one employed in this paper. It is worth noting that the cumulative regret in these papers is defined in a weaker form compared to the traditional bandit literature, since the cumulative reward of a proposed policy is compared to that of the best policy in a certain restricted class of policies. Therefore, bounds on the regret depend, among other things, on the complexity of said class of policies. Plug-in type policies have received attention in the context of the continuum armed bandit problem, where as the name suggests there are uncountably many arms. Notable entries in that stream of work are Slivkins (2009) and Lu et al. (2009), who impose a smoothness condition both on the space of arms and the space of covariates, obtaining optimal regret bounds up to logarithmic terms.

The second key assumption in our paper is a so-called *margin condition*, as it has been come be known in the full information setup; cf. Tsybakov (2004). In that setting, it has been shown to critically affect the complexity of classification problems (Tsybakov, 2004; Boucheron et al., 2005; Audibert & Tsybakov, 2007). In the bandit setup, this condition encodes the "separation" between the functions that describe the arms' responses and was originally studied by Goldenshluger and Zeevi (2009) in the one armed bandit problem; see further discussion in section 2. We will see later that the margin condition is a natural measure of complexity in the nonparametric bandit problem.

In this paper, we introduce a family of policies called UCBograms. The term is indicative of two salient ingredients of said policies: they build on *regressogram* estimators; and augment the resulting mean response estimates with upper-confidence-bound terms. The idea of the regressogram is quite natural and easy to implement. It groups the covariate vectors into bins and then estimates, by means of simple averaging, a constant which is a proxy for the mean response of each arm over each such bin. One then views these bins as indexing "local" bandit problems, which are solved by applying a suitable UCB-type modification, following the logic of Lai and Robbins (1985) and Auer et al. (2002). In other words, this family of policies decomposes the non-parametric bandit problem into a sequence of localized standard bandit problems; see section 3 for a complete description. The idea of binning covariates lends itself to natural implementation in the two motivating examples described earlier: patients and consumers are segmented into groups with "similar" characteristics; and then the treatment or ad is allocated based on the characteristic response over that group.

In terms of performance, we prove that the UCBogram policies achieve a regret that is fairly large compared to typical orders of regret observed in the literature. In particular, as opposed to a bounded or logarithmic growth, in our setting the order of the regret is *polynomial* in the time horizon n ; see Theorem 3.1. One may question, especially given the simple structure and logic underlying the UCBogram policy, whether this is the best that can be achieved in such problems. To that end, we prove a lower bound which demonstrates that for any admissible policy there exist arm response functions satisfying our assumptions for which one cannot improve on the polynomial order of the upper bound established in Theorem 3.1; see Theorem 4.1. Finally, beyond these analytical results, in our view one of the contributions of the present paper is in pointing to some possible synergies and potentially interesting connections between the traditional bandit literature

and nonparametric statistics.

2 Description of the problem

2.1 Machine and game

A *bandit machine with covariates* is characterized by a sequence

$$(X_t, Y_t^{(1)}, Y_t^{(2)}), t = 1, 2, \dots$$

of independent random vectors, where $(X_t), t = 1, 2, \dots$ is a sequence of iid covariates in $\mathcal{X} \subset \mathbb{R}^d$ with probability distribution P_X , and $Y_t^{(i)}$ denotes the random reward yielded by arm i at time t . Denote by E_X the expectation with respect to P_X . We assume that, for each $i = 1, 2$, the rewards $Y_t^{(i)}, t = 1, \dots, n$ are random variables in $[0, 1]$ with conditional expectation given by

$$\mathbb{E}[Y_t^{(i)} | X_t] = f^{(i)}(X_t), \quad t = 1, 2, \dots, i = 1, 2,$$

where $f^{(i)}, i = 1, 2$, are unknown functions such that $0 \leq f^{(i)}(x) \leq 1$, for any $i = 1, 2, x \in \mathcal{X}$. A natural example is a where $Y_t^{(i)}$ takes values in $\{0, 1\}$ so that the conditional distribution of $Y_t^{(i)}$ given X_t is Bernoulli with parameter $f^{(i)}(X_t)$.

The *game* takes place sequentially on this machine, pulling one of the two arms at each time $t = 1, \dots, n$. A *non-anticipating policy* $\pi = \{\pi_t\}$ is a sequence of random functions $\pi_t : \mathcal{X} \rightarrow \{1, 2\}$ indicating to the operator which arm to pull at each time t , and such that π_t depends only on observations strictly anterior to t . The *oracle rule* π^* , refers to the strategy that would be played by an omniscient operator with complete knowledge of the functions $f^{(i)}, i = 1, 2$. Given side information X_t , the oracle policy π^* prescribes the arm with the largest expected reward, i.e.,

$$\pi^*(X_t) \in \arg \max_{i=1,2} f^{(i)}(X_t).$$

The oracle rule will be used to benchmark any proposed policy π and to measure the performance of the latter via its (*cumulative*) *regret* at time n defined by

$$R_n(\pi) := \mathbb{E} \sum_{t=1}^n (Y_t^{\pi^*(X_t)} - Y_t^{\pi_t(X_t)}) = \sum_{t=1}^n E_X (f^{\pi^*(X)}(X) - f^{\pi_t(X)}(X)).$$

Also, let $S_n(\pi)$ denote the *inferior sampling rate* at time n defined by

$$S_n(\pi) := \sum_{t=1}^n P_X(\pi_t(X) \neq \pi^*(X), f^{(1)}(X) \neq f^{(2)}(X)), \quad (1)$$

where $\mathbb{I}(A)$ is the indicator function that takes value 1 if event A is realized and 0 otherwise. The quantity $S_n(\pi)$ measures the expected number of times at which a *strictly* suboptimal arm has been pulled, and note that in our setting the suboptimal arm varies as a function of the covariate value x .

Without further assumptions on the machine, the game can be arbitrarily difficult and, as a result, the regret and inferior sampling rate can be arbitrarily close to n . In the following subsection, we describe natural regularity conditions under which it is possible to achieve sublinear growth rate of the regret and inferior sampling rate, and characterize policies that perform in a near-optimal manner.

2.2 Smoothness and margin conditions

As usual in nonparametric estimation we first impose some regularity on the functions $f^{(i)}, i = 1, 2$. Here and in what follows we use $\|\cdot\|$ to denote the Euclidean norm.

SMOOTHNESS CONDITION. We say that the machine satisfies the smoothness condition with parameters (β, L) if

$$|f^{(i)}(x) - f^{(i)}(x')| \leq L \|x - x'\|^\beta, \quad \forall x, x' \in \mathcal{X}, i = 1, 2 \quad (2)$$

for some $\beta \in (0, 1]$ and $L > 0$.

Notice that a direct consequence of the smoothness condition with parameters (β, L) is that the function $\Delta := |f^{(1)} - f^{(2)}|$ also satisfies the smoothness condition with parameters $(\beta, 2L)$. The behavior of function Δ critically controls the complexity of the problem and the smoothness condition gives a local upper bound on this quantity. The second condition imposed gives a lower bound on this function though in a weaker global

sense. It is closely related to the margin condition employed in classification (Tsybakov, 2004; Mammen & Tsybakov, 1999), which drives the terminology employed here.

MARGIN CONDITION. We say that the machine satisfies the margin condition with parameter α if there exists $\delta_0 \in (0, 1)$, $C_\delta > 0$ such that

$$P_X [0 < |f^{(1)}(X) - f^{(2)}(X)| \leq \delta] \leq C_\delta \delta^\alpha, \quad \forall \delta \in [0, \delta_0]$$

for some $\alpha > 0$.

In what follows, we will focus our attention on marginals P_X that are equivalent to the Lebesgue measure on a compact subset of \mathbb{R}^d . In that way, the margin condition will only contain information about the behavior of the function Δ and not the marginal P_X itself. A large value of the parameter α means that the function Δ either takes value 0 or is bounded away from 0, except over a set of small P_X -probability. Conversely, for values of α close to 0, the margin condition is essentially void and the two functions can be arbitrary close, making it difficult to distinguish them. This will be reflected in the bounds on the regret which are derived in the subsequent section.

Intuitively, the smoothness condition and the margin condition work in opposite directions. Indeed, the former ensures that the function Δ does not “depart from zero” too fast whereas the latter warrants the opposite. The following proposition quantifies the extent to which the conditions are conflicting.

Proposition 2.1 *Under the smoothness condition with parameters (β, L) , any machine that satisfies the margin condition with parameter α such that $\alpha\beta > 1$ exhibits an oracle policy π^* which dictates pulling only one of the two arms all the time, P_X -almost surely. Conversely, if $\alpha\beta \leq 1$ there exist machines with nontrivial oracle policies.*

Proof. The first part of the proof is a straightforward consequence of Proposition 3.4 in Audibert and Tsybakov (2007). To prove the second part, consider the following example. Assume that $d = 1$, $\mathcal{X} = [0, 2]$, $f^{(2)} \equiv 0$ and $f^{(1)}(x) = L \text{sign}(x - 1)|x - 1|^{1/\alpha}$. Notice that $f^{(1)}$ satisfies the smoothness condition with parameters (β, L) if and only if $\alpha\beta \leq 1$. The oracle policy is not trivial and defined by $\pi^*(x) = 2$ if $x \leq 1$ and $\pi^*(x) = 1$ if $x > 1$. Moreover, it can be easily shown that the machine satisfies the margin condition with parameter α and with $\delta_0 = C_\delta = 1$. ■

3 Policy and main result

We first outline a policy to operate the bandit machine described in the previous section. Then we state the main result which is an upper bound on the regret for this policy. Finally, we state a proposition which allows us to translate the bound on the regret into a bound on the inferior sampling rate.

3.1 Binning and regressograms

To design a policy that solves the bandit problem described in the previous section, one has to inevitably find an estimate of the functions $f^{(i)}$, $i = 1, 2$ at the current point X_t . There exists a wide variety of non-parametric regression estimators ranging from local polynomials to wavelet estimators. However, a very simple piecewise constant estimator, commonly referred to as *regressogram* will be particularly suitable for our purposes.

Assume now that $\mathcal{X} = [0, 1]^d$ and let $\{B_j, j = 1, \dots, M^d\}$ be a regular covering of \mathcal{X} , i.e., the reindexed collection of hypercubes defined for $k = (k_1, \dots, k_d) \in \{1, \dots, M\}^d$ by

$$B_k = \left\{ x \in \mathcal{X} : \frac{k_\ell - 1}{M} \leq x_\ell \leq \frac{k_\ell}{M}, \ell = 1, \dots, d \right\}.$$

As stated earlier, we assume that P_X is absolutely continuous with respect to the Lebesgue measure, so that for any $\mathcal{I} \subset \{1, \dots, M^d\}$ we have $P_X(\bigcap_{j \in \mathcal{I}} B_j) = 0$. As a result, we will omit from our analysis considerations about events where $X \in \bigcap_{j \in \mathcal{I}} B_j$ for some $\mathcal{I} \subset \{1, \dots, M^d\}$.

For each arm $i = 1, 2$, consider the average reward for each bin $B_j, j = 1, \dots, M^d$ defined by

$$\bar{f}_j^{(i)} = \mathbb{E}[f^{(i)}(X_t) | X_t \in B_j] = \frac{1}{p_j} \int_{B_j} f^{(i)}(x) dP_X(x),$$

where $p_j = P_X(B_j)$. By analogy with histograms, the empirical counterpart of the piecewise constant function $x \mapsto \sum_{j=1}^{M^d} \bar{f}_j^{(i)} \mathbb{I}(x \in B_j)$, is often called a *regressogram*. To define it, we need the following

quantities. Let $N_t^{(i)}(j, \pi)$ denote the number of times π prescribes to pull arm i at times anterior to t when the covariate was in bin B_j ,

$$N_t^{(i)}(j, \pi) = \sum_{s=1}^t \mathbb{I}(X_s \in B_j, \pi_s(X_s) = i),$$

and let $\bar{Y}_t^{(i)}(j, \pi)$ denote the average reward collected at those times,

$$\bar{Y}_t^{(i)}(j, \pi) = \frac{1}{N_t^{(i)}(j, \pi)} \sum_{s=1}^t Y_s^{(i)} \mathbb{I}(X_s \in B_j, \pi_s(X_s) = i),$$

where here and throughout this paper, we use the convention $1/0 = \infty$. For any arm $i = 1, 2$ and any time $t \geq 1$ the regressograms obtained from a policy π at time t are defined by the following piecewise constant estimators

$$\hat{f}_{t,\pi}^{(i)}(x) = \sum_{j=1}^{M^d} \bar{Y}_t^{(i)}(j, \pi) \mathbb{I}(x \in B_j).$$

While regressograms are rather rudimentary nonparametric estimators of the functions $f^{(i)}$, they allow us to decompose the original problem into a collection of M^d traditional bandit machines without covariates, each one corresponding to a different bin.

3.2 The UCBogram

The ‘‘UCBogram’’ is an index type policy based on upper confidence bounds for the regressogram defined above. Upper confidence bounds (UCB) policies are known to perform optimally in the traditional two armed bandit problem, i.e., without covariates (Lai & Robbins, 1985; Auer et al., 2002). The index of each arm is computed as the sum of the average past reward and a stochastic term accounting for the deviations of the observed average reward from the true average reward. In the UCBogram, the average reward is simply replaced by the value of the regressogram at the current covariate X_t .

For any $s \geq 1$ the upper confidence bound at time t bound is of the form

$$U_t(s) = \sqrt{\frac{2 \log t}{s}},$$

and $U_t(0) = 0$. The UCBogram $\hat{\pi}$ is defined as follows. For any $x \in [0, 1]^d$, define

$$N_t^{(i)}(x) = \sum_{j=1}^{M^d} N_t^{(i)}(j, \hat{\pi}) \mathbb{I}(x \in B_j),$$

the number of times the UCBogram prescribes to pull arm i at times anterior to t when the covariate was in the same bin as x . Then $\hat{\pi} = (\hat{\pi}_1, \hat{\pi}_2, \dots)$ is defined recursively by

$$\hat{\pi}_t(x) \in \arg \max_{i=1,2} \left\{ \hat{f}_{t,\hat{\pi}}^{(i)}(x) + U_t(N_t^{(i)}(x)) \right\}.$$

Notice that the UCBogram is indeed a UCB-type policy. Indeed, for each arm $i = 1, 2$ and each point x , it computes an estimator $\hat{f}_{t,\hat{\pi}}^{(i)}(x)$ of the expected reward and adds an upper confidence bound $U_t(N_t^{(i)}(x))$ to account for stochastic variability in this estimator. The most attractive feature of the regressogram is that it allows to decompose the nonparametric bandit problem into independently operated local machines as detailed in the proof of the following theorem.

Theorem 3.1 Fix $\beta \in (0, 1]$, $L > 0$ and $\alpha \in (0, 1]$. Let $\mathcal{X} = [0, 1]^d$ and assume that the covariates X_t have a distribution which is equivalent¹ to the Lebesgue measure on the unit hypercube \mathcal{X} . Let the machine satisfy both the smoothness condition with parameter (β, L) and the margin condition with parameter $0 < \alpha \leq 1$. Then the UCBogram policy $\hat{\pi}$ with $M = \lfloor (n/\log n)^{1/(2\beta+d)} \rfloor$ has an expected cumulative regret at time n bounded as follows,

$$\mathbb{E}R_n(\hat{\pi}) \leq Cn \max \left\{ \left(\frac{n}{\log n} \right)^{-\frac{\beta(\alpha+1)}{2\beta+d}}, \left(\frac{n}{(\log n)^2} \right)^{-\frac{2\beta}{2\beta+d}} \right\},$$

where $C > 0$ is a positive constant.

¹Two measures μ and ν are said to be *equivalent* if there exist two positive constants \underline{c} and \bar{c} such that $\underline{c}\mu(A) \leq \nu(A) \leq \bar{c}\mu(A)$ for any measurable set A .

Proof. To keep track of positive constants, we number them c_1, c_2, \dots . Define $c_1 = 2Ld^{\beta/2} + 1$, and let $n_0 \geq 2$ be the largest integer such that

$$\left(\frac{n_0}{\log n_0} \right)^{\beta/(2\beta+d)} \leq \frac{2c_1}{\delta_0},$$

where δ_0 is the constant appearing in the margin condition. If $n \leq n_0$, we have $R_n(\hat{\pi}) \leq n_0$ so that the result of the theorem holds when C is chosen large enough, depending on the constant n_0 . In the rest of the proof, we assume that $n > n_0$ so that $c_1 M^{-\beta} < \delta_0$.

Recall that the UCBogram policy $\hat{\pi}$ is a collection of functions $\hat{\pi}_t$ that are constant on each B_j . Define the regret $R_j(\hat{\pi})$ on bin B_j by

$$R_j(\hat{\pi}) = \sum_{t=1}^n (f^{(\pi^*(X_t))}(X_t) - f^{(\hat{\pi}_t(x_j))}(X_t)) \mathbb{I}(X_t \in B_j),$$

where x_j is an arbitrary element of B_j . Observe that the overall regret of $\hat{\pi}$ can be written as

$$\mathbb{E}R_n(\hat{\pi}) = \sum_{j=1}^{M^d} \mathbb{E}R_j(\hat{\pi}).$$

Consider the set of ‘‘well behaved’’ bins on which the expected reward functions of the two arms are well separated:

$$\mathcal{J} = \{j : \exists x \in B_j, |f^{(1)}(x) - f^{(2)}(x)| > c_1 M^{-\beta}\}.$$

For any $j \notin \mathcal{J}$ and any $x \in B_j$, we have $|f^{(1)}(x) - f^{(2)}(x)| \leq c_1 M^{-\beta} < \delta_0$ so that

$$\mathbb{E}R_j(\hat{\pi}) \leq c_1 M^{-\beta} \sum_{t=1}^n \mathbb{P}[0 < |f^{(1)}(X_t) - f^{(2)}(X_t)| \leq c_1 M^{-\beta}, X_t \in B_j],$$

Summing over $j \notin \mathcal{J}$, we obtain from the margin condition that

$$\sum_{j \notin \mathcal{J}} \mathbb{E}R_j(\hat{\pi}) \leq C_\delta c_1^{1+\alpha} n M^{-\beta(1+\alpha)}. \quad (3)$$

We now treat the well behaved bins, i.e., bins B_j such that $j \in \mathcal{J}$. Notice that since each bin is a hypercube with side length $1/M$ and since the reward functions satisfy the smoothness condition with parameters (β, L) , we have

$$|f^{(1)}(x) - f^{(2)}(x)| > c_1 M^{-\beta} - 2Ld^{\beta/2} M^{-\beta} = M^{-\beta},$$

for any $x \in B_j, j \in \mathcal{J}$. In particular, for such j , since the two functions are continuous, the difference $f^{(1)}(x) - f^{(2)}(x)$ has constant sign over B_j and $|\bar{f}_j^{(1)} - \bar{f}_j^{(2)}| > M^{-\beta}$. As a consequence, the oracle policy π^* is constant on B_j , equal to $\pi^*(j)$ for any $j \in \mathcal{J}$ and, conditionally on $\{X_t \in B_j\}$, the game can be viewed as a standard bandit problem, i.e., without covariates, where arm i has bounded reward with mean $\bar{f}_j^{(i)}$. Moreover, conditionally on $\{X_t \in B_j\}$, the UCBogram can be seen as a standard UCB policy. Applying for example Theorem 1 in Auer et al. (2002), we find that for $j \in \mathcal{J}$,

$$\mathbb{E}R_j(\hat{\pi}) \leq \left[\left(1 + \frac{\pi^2}{3}\right) \Delta_j \right] + \frac{8 \log n}{\Delta_j} \leq c_2 \frac{\log n}{\Delta_j}, \quad (4)$$

where $\Delta_j = |\bar{f}_j^{(1)} - \bar{f}_j^{(2)}|$ is the average gap over the bin B_j . Note that the UCB policy employed here uses the term $\log t$ instead of $\log(N_t^{(1)}(j, \pi) + N_t^{(2)}(j, \pi))$ which is prescribed in Auer et al. (2002); it is easy to verify that either choice leads to an identical bound on the regret.

We now use the margin condition to provide lower bounds on Δ_j . Assume without loss of generality that the gaps are ordered $0 < \Delta_1 \leq \Delta_2 \leq \dots \leq \Delta_{M^d}$ and define the integers $j_1 = \min(\mathcal{J})$ and $j_2 \in \{j_1, \dots, M^d\}$ to be the largest integer such that $\Delta_{j_2} \leq \delta_0/c_1$. Therefore, for any $j \in \{j_1, \dots, j_2\} \cap \mathcal{J}$, we have on the one hand,

$$P_X[0 < |f^{(1)} - f^{(2)}| \leq \Delta_j + (c_1 - 1)M^{-\beta}] \geq \sum_{k=1}^{M^d} p_k \mathbb{I}(0 < \Delta_k \leq \Delta_j) \geq \frac{c_j}{M^d}, \quad (5)$$

where we use the fact that $p_k = P_X(B_k) \geq c/M^d$ since P_X is equivalent to the Lebesgue measure on $[0, 1]^d$ (see footnote 1). On the other hand, the margin condition yields for any $j \in \{j_1, \dots, j_2\} \cap \mathcal{J}$ that,

$$P_X[0 < |f^{(1)} - f^{(2)}| \leq \Delta_j + (c_1 - 1)M^{-\beta}] \leq C_\delta (c_1 \Delta_j)^\alpha, \quad (6)$$

where we have used the fact that $\Delta_j + (c_1 - 1)M^{-\beta} \leq c_1 \Delta_j \leq \delta_0$, for any $j \in \{j_1, \dots, j_2\} \cap \mathcal{J}$. The previous two inequalities together with the fact that $\Delta_j > M^{-\beta}$ for any $j \in \mathcal{J}$, yield

$$\Delta_j \geq c_3 \left(\frac{j}{M^d}\right)^{1/\alpha} \vee M^{-\beta}, \quad \forall j \in \{j_1, \dots, j_2\} \cap \mathcal{J}. \quad (7)$$

Combining (3), (4) and (7), we obtain the following bound,

$$\mathbb{E}R_n(\hat{\pi}) \leq c_4 \left[nM^{-\beta(1+\alpha)} + (\log n) \sum_{j=j_1}^{j_2} \left[\left(\frac{M^d}{j}\right)^{\frac{1}{\alpha}} \wedge M^\beta \right] + M^d \log n \right]. \quad (8)$$

We now bound from above the sum in (8). Note that

$$\sum_{j=j_1}^{j_2} \left[\left(\frac{M^d}{j}\right)^{\frac{1}{\alpha}} \wedge M^\beta \right] \leq \sum_{j=1}^{M^d} \left[\left(\frac{M^d}{j}\right)^{\frac{1}{\alpha}} \wedge M^\beta \right] \leq c_5 \left[M^{d+\beta(1-\alpha)} + \sum_{j=M'+1}^{M^d} \left(\frac{M^d}{j}\right)^{\frac{1}{\alpha}} \right], \quad (9)$$

where $M' = c_6 M^{d-\alpha\beta}$. Moreover,

$$\sum_{j=M'+1}^{M^d} \left(\frac{M^d}{j}\right)^{\frac{1}{\alpha}} \leq c_6 M^d \int_{M^{-\alpha\beta}}^1 x^{-1/\alpha} dx.$$

If $\alpha < 1$, this integral is bounded by $c_7 M^{\beta(1-\alpha)}$ and if $\alpha = 1$, it is bounded by $c_8 \log M$. As a result, the right hand side of (9) is of order $M^d (M^{\beta(1-\alpha)} \vee \log M)$ and we obtain from (8) that

$$\mathbb{E}R_n(\hat{\pi}) \leq c_9 \left[nM^{-\beta(1+\alpha)} + M^d (M^{\beta(1-\alpha)} \vee \log M) \log n \right], \quad (10)$$

and the result follows by choosing M as prescribed. ■

We should point out that the version of the UCBogram described above specifies the number of bins M as a function of the horizon n , while in practice one does not have foreknowledge of this value. This limitation can be easily circumvented by using the so-called *doubling argument* (Cesa-Bianchi & Lugosi, 2006) which consists of “resetting” the game at times $2^k, k = 1, 2, \dots$.

The reader will note that when $\alpha = 1$ there is a potentially superfluous $\log n$ factor appearing in the upper bound in the theorem. More generally, for any $\alpha > 1$, it is possible to minimize the expression on the right hand side of (10) with respect to M , but the optimal value of M would then depend on the value of α . This sheds some light on a significant limitation of the UCBogram which surfaces in this parameter regime: it requires the operator to pull each arm at least once in each bin and therefore to incur a regret of at least order M^d . In other words, the UCBogram splits the space \mathcal{X} in “too many” bins when $\alpha \geq 1$. Intuitively this can be understood as follows. When $\alpha = 1$, the gap function $\Delta(x)$ is bounded away from zero for most $x \in \mathcal{X}$, and hence there is no need to carefully estimate the gap function since it has constant sign over “large” contiguous regions. As a result one could use larger bins in such regions reducing the overall number of bins and therefore removing the extra logarithmic term alluded to above. These limitations are obviously intrinsic to UCBogram-type policies.

3.3 The inferior sampling rate

Unlike traditional bandit problems, the connection between the inferior sampling rate defined in (1) and the regret is more intricate here. The following lemma establishes a connection between the two.

Lemma 3.1 *For any $\alpha > 0$, under the margin condition we have*

$$S_n(\pi) \leq C n^{\frac{1}{1+\alpha}} R_n(\pi)^{\frac{\alpha}{1+\alpha}},$$

for any policy π and for some positive constant $C > 0$.

Proof. The idea of the proof is quite standard and originally appeared in Tsybakov (2004). It has been used in Rigollet and Vert (2009) and Goldenshluger and Zeevi (2009). Define the two random quantities:

$$r_n(\pi) = \sum_{t=1}^n |f^{(1)}(X_t) - f^{(2)}(X_t)| \mathbb{I}(\pi_t(X_t) \neq \pi^*(X_t)),$$

and

$$s_n(\pi) = \sum_{t=1}^n \mathbb{I}(f^{(1)}(X_t) \neq f^{(2)}(X_t), \pi_t(X_t) \neq \pi^*(X_t)).$$

We have

$$\begin{aligned} r_n(\pi) &\geq \delta \sum_{t=1}^n \mathbb{I}(\pi_t(X_t) \neq \pi^*(X_t)) \mathbb{I}(|f^{(1)}(X_t) - f^{(2)}(X_t)| > \delta) \\ &\geq \delta \left[s_n(\pi) - \sum_{t=1}^n \mathbb{I}(\pi_t(X_t) \neq \pi^*(X_t), 0 < |f^{(1)}(X_t) - f^{(2)}(X_t)| \leq \delta) \right] \\ &\geq \delta \left[s_n(\pi) - \sum_{t=1}^n \mathbb{I}(0 < |f^{(1)}(X_t) - f^{(2)}(X_t)| \leq \delta) \right]. \end{aligned} \quad (11)$$

Taking expectations on both sides of (11), we obtain that $R_n(\pi) \geq \delta [S_n(\pi) - n\delta^\alpha]$, where we used the margin condition. The proof follows by choosing $\delta = (S_n(\pi)/cn)^{1/\alpha}$ for $c \geq 2$ large enough to ensure that $\delta < \delta_0$ \blacksquare

Using Lemma 3.1, and the we obtain the following corollary of Theorem 3.1.

Corollary 3.1 Fix $\beta \in (0, 1]$, $L > 0$ and $\alpha \in (0, 1]$. Under the conditions of Theorem 3.1, the UCBogram policy $\hat{\pi}$ with $M = \lfloor (n/\log n)^{1/(2\beta+d)} \rfloor$ has an inferior sampling rate at time n bounded as follows,

$$\mathbb{E}S_n(\hat{\pi}) \leq Cn \max \left\{ \left(\frac{n}{\log n} \right)^{-\frac{\beta\alpha}{2\beta+d}}, \left(\frac{n}{(\log n)^2} \right)^{-\frac{\beta}{2\beta+d}} \right\}.$$

where $C > 0$ is a positive constant.

Proof. Since the function $x \mapsto x^{\frac{\alpha}{1+\alpha}}$ is concave for any $\alpha > 0$, we can apply the Jensen inequality to the result of Lemma 3.1 to obtain that

$$\mathbb{E}S_n(\hat{\pi}) \leq Cn^{\frac{1}{1+\alpha}} [\mathbb{E}R_n(\pi)]^{\frac{\alpha}{1+\alpha}}.$$

The conclusion follows by bounding $\mathbb{E}R_n(\pi)$ from above using Theorem 3.1. \blacksquare

4 Lower bound

While the UCBogram is a very simple policy, it still provides good insights as to how to construct a lower bound on the regret incurred by any admissible policy. Indeed, the main result of this section demonstrates that the polynomial rate of the upper bounds in Theorem 3.1 and Corollary 3.1 is optimal in a minimax sense, for a large class of conditional reward distributions. Define the Kullback-Leibler (KL) divergence between P and Q , where P and Q are two probability distributions by

$$\mathcal{K}(P, Q) = \begin{cases} \int \log \left(\frac{dP}{dQ} \right) dP & \text{if } P \ll Q, \\ \infty & \text{otherwise.} \end{cases}$$

Denote by $P_{f(X)}^{(i)}$ the conditional distribution of $Y^{(i)}$ given X for any $i = 1, 2$ and assume that there exists $\kappa^2 > 0$ such that for any $\theta, \theta' \in \Theta \subset [0, 1]$ the KL divergence between $P_\theta^{(i)}$ and $P_{\theta'}^{(i)}$ satisfies

$$\mathcal{K}(P_\theta^{(i)}, P_{\theta'}^{(i)}) \leq \frac{1}{\kappa^2} (\theta - \theta')^2. \quad (12)$$

Assumption (12) is similar to Assumption (B) employed in Tsybakov, (2009, Section 2.5) but does not require absolute continuity with respect to the Lebesgue measure. A direct consequence of the following lemma is that Assumption (12) is satisfied when P_θ is a Bernoulli distribution with parameter $\theta \in (0, 1)$.

Lemma 4.1 For any $a \in [0, 1]$ and $b \in (0, 1)$ let P_a and P_b denote two Bernoulli distributions with parameters a and b respectively. Then

$$\mathcal{K}(P_a, P_b) \leq \frac{(a-b)^2}{b(1-b)}.$$

In particular, if $b_0 \in [0, 1/2)$, Assumption (12) is satisfied with $\kappa^2 = 1/4 - b_0^2$, for any $a \in [0, 1], b \in [1/2 - b_0, 1/2 + b_0]$.

Proof. From the definition of the KL divergence, we have

$$\mathcal{K}(P_a, P_b) = a \log\left(\frac{a}{b}\right) + (1-a) \log\left(\frac{1-a}{1-b}\right) \leq a\left(\frac{a-b}{b}\right) - (1-a)\left(\frac{a-b}{1-b}\right) = \frac{(a-b)^2}{b(1-b)}.$$

■

Theorem 4.1 Fix $\alpha, \beta, L > 0$ such that $\alpha\beta < 1$ and let $\mathcal{X} = [0, 1]^d$. Assume that the covariates X_t are uniformly distributed on the unit hypercube \mathcal{X} and that there exists $\tau \in (0, 1/2)$ such that $\{P_\theta^{(i)}, \theta \in [1/2 - \tau, 1/2 + \tau]\}$ satisfies equation (12) for $i = 1, 2$. Then, there exists a pair of reward functions $f^{(i)}, i = 1, 2$ that satisfy both the smoothness condition with parameters (β, L) and the margin condition with parameter α , such that for any non-anticipating policy π the regret is bounded as follows

$$\mathbb{E}R_n(\pi) \geq Cn^{1-\frac{\beta(\alpha+1)}{2\beta+d}}, \quad (13)$$

and the inferior sampling rate is bounded as follows

$$\mathbb{E}S_n(\pi) \geq Cn^{1-\frac{\beta\alpha}{2\beta+d}}, \quad (14)$$

for some positive constant C .

Proof. To simplify the arguments below, it will be useful to denote arm 2 by -1 . Finally, with slight abuse of notation, we use $S_n(\pi, f^{(1)}, f^{(-1)})$ to denote the inferior sampling rate at time n that is defined in (1), making the dependence on the mean reward functions explicit.

In view of Lemma 3.1, it is sufficient to prove (14). To do so we reduce our problem to a hypothesis testing problems; an approach this is quite standard in the nonparametric literature, cf. (Tsybakov, 2009, Chapter 2). For any policy π , and any $t = 1, \dots, n$, denote by $\mathbb{P}_{\pi, f}^t$ the joint distribution of the collection of pairs

$$(X_1, Y_1^{\pi_1(X_1)}), \dots, (X_t, Y_t^{\pi_t(X_t)})$$

where $\mathbb{E}[Y^{(1)}|X] = f(X)$ and $\mathbb{E}[Y^{(-1)}|X] = 1/2$. Let $\mathbb{E}_{\pi, f}^t$ denote the corresponding expectation. It follows that the oracle policy π_f^* is given by $\pi_f^*(x) = \text{sign}[f(x) - 1/2]$ with the convention that $\text{sign}(0) = 1$. Fix $\delta_0 \in (0, 1)$ as in the definition of the margin condition. We now construct a class \mathcal{C} of functions $f : \mathcal{X} \rightarrow [0, 1]$ such that f satisfies (2) and

$$P_X[0 < |f(X) - 1/2| \leq \delta] \leq C_\delta \delta^\alpha, \quad \forall \delta \in [0, \delta_0],$$

As a result, the machine characterized by the expected rewards $f^{(1)} = f$ and $f^{(-1)} = 1/2$ satisfies both the smoothness and the margin conditions. Moreover, we construct \mathcal{C} in such a way that for any policy π

$$\sup_{f \in \mathcal{C}} \mathbb{E}S_n(\pi, f, 1/2) \geq Cn \left(\frac{n}{\log n}\right)^{-\frac{\beta\alpha}{2\beta+d}}. \quad (15)$$

for some positive constant C . Consider the regular grid $\mathcal{Q} = \{q_1, \dots, q_{M^d}\}$, where q_k denotes the center of bin B_k , $k = 1, \dots, M^d$, for some $M \geq 1$ to be defined. Define $C_\phi = \min(2^{\beta-1}L, \tau, 1/4)$ and let $\phi : \mathbb{R}^d \rightarrow \mathbb{R}_+$ be the function defined by

$$\phi(x) = \begin{cases} (1 - \|x\|_\infty)^\beta & \text{if } 0 \leq \|x\|_\infty \leq 1, \\ 0 & \text{if } \|x\|_\infty > 1. \end{cases}$$

Clearly, we have $|C_\phi \phi(x) - C_\phi \phi(x')| \leq 2^{\beta-1}L\|x - x'\|_\infty^\beta \leq 2^{\beta-1}L\|x - x'\|^\beta$ for any $x, x' \in \mathbb{R}^d$.

Define the integer $m = \lceil \mu M^{d-\alpha\beta} \rceil$, i.e., the smallest integer that is larger than or equal to $\mu M^{d-\alpha\beta}$, where $\mu \in (0, 1)$ is chosen small enough to ensure that $m \leq M^d$. Define $\Omega_m = \{-1, 1\}^m$ and for any $\omega \in \Omega_m$, define the function f_ω on $[0, 1]^d$ by

$$f_\omega(x) = 1/2 + \sum_{j=1}^m \omega_j \varphi_j(x),$$

where $\varphi_j(x) = M^{-\beta} C_\phi \phi(M[x - q_j]) \mathbb{I}(x \in B_j)$. Notice in particular that $f_\omega(x) = 1/2$ if and only if $x \in \mathcal{X} \setminus \bigcup_{j=1}^m B_j$ up to a set of zero Lebesgue measure. We are now in position to define the family \mathcal{C} as

$$\mathcal{C} = \{f_\omega : \omega \in \Omega_m\}.$$

Note first that any function $f_\omega \in \mathcal{C}$ satisfies the smoothness condition (2). Indeed, if u and v belong to the same bin B_j , then

$$|f_\omega(u) - f_\omega(v)| \leq |\varphi_j(u) - \varphi_j(v)| \leq 2^{\beta-1} L \|u - v\|^\beta \leq L \|u - v\|^\beta. \quad (16)$$

If $u \in B_j$ and $v \in B_k$, $j \neq k$, consider the segment $S_{u,v} = \{\theta u + (1 - \theta)v : \theta \in [0, 1]\}$ between u and v and define the points

$$u' = \operatorname{argmin}_{z \in S_{u,v} \cap B_j} \|z - v\|, \quad v' = \operatorname{argmin}_{z \in S_{u,v} \cap B_k} \|z - u\|.$$

We have $u' \in B_j$, $v' \in B_k$, and $\varphi_j(u') = \varphi_k(v') = 0$ so that

$$\begin{aligned} |f_\omega(u) - f_\omega(v)| &\leq |\varphi_j(u) - \varphi_j(u')| + |\varphi_k(v) - \varphi_k(v')| \\ &\leq 2^{\beta-1} L \|u - u'\|^\beta + 2^{\beta-1} L \|v - v'\|^\beta \\ &\leq L \|u - v\|^\beta, \end{aligned}$$

where in the second inequality we used (16) and in the third, we used the concavity of the function $x \mapsto x^\beta$ for $\beta \leq 1$ together with the fact that $\|u - u'\| + \|v - v'\| \leq \|u - v\|$.

We now check that the margin condition is satisfied with parameter α . For any $\omega \in \Omega_m$, we have

$$\begin{aligned} P_X(0 < |f_\omega(X) - 1/2| \leq C_\phi \delta) &= \sum_{j=1}^m P_X(0 < |f_\omega(X) - 1/2| \leq C_\phi \delta, X \in B_j) \\ &= m P_X(0 < \phi(M[X - q_1]) \leq \delta M^\beta, X \in B_1) \\ &= m \int_{B_1} \mathbb{I}(\phi(Mx) \leq \delta M^\beta) dx \\ &= m M^{-d} \int_{[0,1]^d} \mathbb{I}(\phi(x) \leq \delta M^\beta) dx, \end{aligned}$$

where in the third equality, we used the fact that P_X denotes the uniform distribution on $[0, 1]^d$. Now, since ϕ is non negative and uniformly bounded by 1, we have on the one hand that for $\delta M^\beta > 1$,

$$\int_{[0,1]^d} \mathbb{I}(\phi(x) \leq \delta M^\beta) dx = 1.$$

On the other hand, when $\delta M^\beta \leq 1$, we find

$$\int_{[0,1]^d} \mathbb{I}(\phi(x) \leq \delta M^\beta) dx = 1 - \int_{[0,1]^d} \mathbb{I}(\|x\|_\infty \leq 1 - M\delta^{1/\beta}) dx = 1 - \left(1 - M\delta^{1/\beta}\right)^d \leq dM\delta^{1/\beta}.$$

It yields

$$\begin{aligned} P_X(0 < |f_\omega(X) - 1/2| \leq C_\phi \delta) &\leq m M^{-d} \mathbb{I}(\delta M^\beta > 1) + m d M^{1-d} \delta^{1/\beta} \mathbb{I}(\delta M^\beta \leq 1) \\ &\leq M^{-\alpha\beta} \mathbb{I}(M^{-\alpha\beta} < \delta^\alpha) + d M^{1-\alpha\beta} \delta^{1/\beta} \mathbb{I}(M \leq \delta^{-1/\beta}) \\ &\leq (1 + d) \delta^\alpha, \end{aligned}$$

where we used the fact that $1 - \alpha\beta \geq 0$ to bound the second term in the last inequality. Thus, the margin condition is satisfied for any δ_0 and with $C_\delta = (1 + d)/C_\phi^\alpha$.

We now prove (15) by observing that if we denote $\omega = (\omega_1, \dots, \omega_m) \in \Omega_m$, we have

$$\begin{aligned} \sup_{f \in \mathcal{C}} \mathbb{E} S_n(\pi, f^{(1)}, 1/2) &= \sup_{\omega \in \Omega_m} \sum_{t=1}^n \mathbb{E}_{\pi, f_\omega}^{t-1} P_X [\pi_t(X_t) \neq \operatorname{sign}(f_\omega(X_t))] \\ &= \sup_{\omega \in \Omega_m} \sum_{j=1}^m \sum_{t=1}^n \mathbb{E}_{\pi, f_\omega}^{t-1} P_X [\pi_t(X_t) \neq \omega_j, X_t \in B_j] \\ &\geq \frac{1}{2m} \sum_{j=1}^m \sum_{t=1}^n \sum_{\omega \in \Omega_m} \mathbb{E}_{\pi, f_\omega}^{t-1} P_X [\pi_t(X_t) \neq \omega_j, X_t \in B_j] \quad (17) \end{aligned}$$

Observe now that for any $j = 1, \dots, m$, the sum $\sum_{\omega \in \Omega} [\dots]$ in the previous display can be decomposed as

$$Q_j^t = \sum_{\omega_{[-j]} \in \Omega_{m-1}} \sum_{i \in \{-1, 1\}} \mathbb{E}_{\pi, f_{\omega_{[-j]}^i}}^{t-1} P_X [\pi_t(X_t) \neq i, X_t \in B_j],$$

where $\omega_{[-j]} = (\omega_1, \dots, \omega_{j-1}, \omega_{j+1}, \dots, \omega_m)$ and $\omega_{[-j]}^i = (\omega_1, \dots, \omega_{j-1}, i, \omega_{j+1}, \dots, \omega_m)$ for $i = -1, 1$. Using Theorem 2.2(iii) of Tsybakov (2009), and denoting by $P_X^j(\cdot)$ the conditional distribution $P_X(\cdot | X \in B_j)$, we get

$$\begin{aligned} \sum_{i \in \{-1, 1\}} \mathbb{E}_{\pi, f_{\omega_{[-j]}^i}}^{t-1} P_X [\pi_t(X_t) \neq i, X_t \in B_j] &= \frac{1}{M^d} \sum_{i \in \{-1, 1\}} \mathbb{E}_{\pi, f_{\omega_{[-j]}^i}}^{t-1} P_X^j [\pi_t(X_t) \neq i] \\ &\geq \frac{1}{4M^d} \exp \left[-\mathcal{K}(\mathbb{P}_{\pi, f_{\omega_{[-j]}^{-1}}}^{t-1} \times P_X^j, \mathbb{P}_{\pi, f_{\omega_{[-j]}^1}}^{t-1} \times P_X^j) \right] \\ &= \frac{1}{4M^d} \exp \left[-\mathcal{K}(\mathbb{P}_{\pi, f_{\omega_{[-j]}^{-1}}}^{t-1}, \mathbb{P}_{\pi, f_{\omega_{[-j]}^1}}^{t-1}) \right] \end{aligned} \quad (18)$$

For any $t = 2, \dots, n$, let \mathcal{F}_t denote the σ -algebra generated by the information available at time t immediately after observing X_t , i.e., $\mathcal{F}_t = \sigma(X_t, (X_s, Y_s^{(\pi_s(X_s))}), s = 1, \dots, t-1)$. Define the conditional distribution $\mathbb{P}_{\pi, f}^{\cdot | \mathcal{F}_t}$ of the random couple $(X_t, Y_t^{(\pi_t(X_t))})$, conditioned on \mathcal{F}_t . Denote also by E_{X_t} the expectation with respect to the marginal distribution of X_t . Applying the chain rule for KL divergence, we find that for any $t = 1, \dots, n$ and any $f, g : \mathcal{X} \rightarrow [0, 1]$, we have

$$\begin{aligned} \mathcal{K}(\mathbb{P}_{\pi, f}^t, \mathbb{P}_{\pi, g}^t) &= \mathcal{K}(\mathbb{P}_{\pi, f}^{t-1}, \mathbb{P}_{\pi, g}^{t-1}) + \mathbb{E}_{\pi, f}^{t-1} E_{X_t} \left[\mathcal{K}(\mathbb{P}_{\pi, f}^{\cdot | \mathcal{F}_t}, \mathbb{P}_{\pi, g}^{\cdot | \mathcal{F}_t}) \right] \\ &= \mathcal{K}(\mathbb{P}_{\pi, f}^{t-1}, \mathbb{P}_{\pi, g}^{t-1}) + \mathbb{E}_{\pi, f}^{t-1} E_{X_t} \left[\mathcal{K}(\mathbb{P}_{\pi, f}^{Y_t^{(\pi_t(X_t))} | \mathcal{F}_t}, \mathbb{P}_{\pi, g}^{Y_t^{(\pi_t(X_t))} | \mathcal{F}_t}) \right], \end{aligned}$$

where $\mathbb{P}_{\pi, f}^{Y_t^{(\pi_t(X_t))} | \mathcal{F}_t}$ denotes the conditional distribution of $Y_t^{(\pi_t(X_t))}$ given \mathcal{F}_t . Since, for any $f \in \mathcal{C}$, we have that $\mathbb{E}[Y_t^{(\pi_t(X_t))} | \mathcal{F}_t] = f^{(\pi_t(X_t))}(X_t) \in [1/2 - \tau, 1/2 + \tau]$, we can apply (12) to derive the following upper bound:

$$\begin{aligned} \mathcal{K}(\mathbb{P}_{\pi, f_{\omega_{[-j]}^{-1}}}^{Y_t^{(\pi_t(X_t))} | \mathcal{F}_t}, \mathbb{P}_{\pi, f_{\omega_{[-j]}^1}^{Y_t^{(\pi_t(X_t))} | \mathcal{F}_t})} &\leq \frac{1}{\kappa^2} \left(f_{\omega_{[-j]}^1}(X_t) - f_{\omega_{[-j]}^{-1}}(X_t) \right)^2 \mathbb{I}(\pi_t(X_t) = 1) \\ &\leq \frac{4}{\kappa^2} C_\phi^2 M^{-2\beta} \mathbb{I}(\pi_t(X_t) = 1, X_t \in B_j) \\ &\leq \frac{M^{-2\beta}}{4\kappa^2} \mathbb{I}(\pi_t(X_t) = 1, X_t \in B_j). \end{aligned}$$

By induction, the last two displays yield that for any $t = 1, \dots, n$,

$$\mathcal{K}(\mathbb{P}_{\pi, f_{\omega_{[-j]}^1}^{t-1}}, \mathbb{P}_{\pi, f_{\omega_{[-j]}^{-1}}^{t-1}}) \leq \frac{M^{-2\beta}}{4\kappa^2} \mathbf{N}_{j, \pi}, \quad (19)$$

where

$$\mathbf{N}_{j, \pi} = \mathbb{E}_{\pi, f_{\omega_{[-j]}^{-1}}^{n-1}} E_X \left[\sum_{t=1}^n \mathbb{I}(\pi_t(X) = 1, X \in B_j) \right],$$

denotes the expected number of times t between time 1 and time n that $X_t \in B_j$ and $\pi_t(X_t) = 1$. Combining (18) and (19), we get

$$Q_j^t \geq \frac{2^{m-1}}{4M^d} \exp \left(-\frac{M^{-2\beta}}{4\kappa^2} \mathbf{N}_{j, \pi} \right). \quad (20)$$

On the other hand, from the definition of Q_j^t , we clearly have

$$\sum_{t=1}^n Q_j^t \geq 2^{m-1} \mathbf{N}_{j, \pi}. \quad (21)$$

Plugging the lower bounds (20) and (21) into (17) yields

$$\begin{aligned} \sup_{f \in \mathcal{C}} \mathbb{E} S_n(\pi, f^{(1)}, 1/2) &\geq \frac{2^{m-1}}{2^m} \sum_{j=1}^m \max \left\{ \frac{n}{4M^d} \exp \left(-\frac{M^{-2\beta}}{4\kappa^2} \mathbf{N}_{j,\pi} \right), \mathbf{N}_{j,\pi} \right\} \\ &\geq \frac{1}{4} \sum_{j=1}^m \left\{ \frac{n}{4M^d} \exp \left(-\frac{M^{-2\beta}}{4\kappa^2} \mathbf{N}_{j,\pi} \right) + \mathbf{N}_{j,\pi} \right\} \\ &\geq \frac{m}{4} \inf_{z \geq 0} \left\{ \frac{n}{4M^d} \exp \left(-\frac{M^{-2\beta}}{4\kappa^2} z \right) + z \right\} \end{aligned}$$

Notice now that

$$z^* = \operatorname{argmin}_{z \geq 0} \left\{ \frac{n}{4M^d} \exp \left(-\frac{M^{-2\beta}}{4\kappa^2} z \right) + z \right\}$$

is strictly positive if and only if $n > 16\kappa^2 M^{2\beta+d}$, in which case

$$z^* = 4\kappa^2 M^{2\beta} \log \left(\frac{n}{16\kappa^2 M^{2\beta+d}} \right).$$

Taking

$$M = \left\lceil \left(\frac{n}{16\kappa^2} \right)^{\frac{1}{2\beta+d}} \right\rceil$$

gives $z^* = c^* n^{\frac{2\beta}{2\beta+d}}$ for some positive constant c^* , so that

$$\sup_{f \in \mathcal{C}} \mathbb{E} S_n(\pi, f^{(1)}, 1/2) \geq C m z^* \geq C n^{1 - \frac{\alpha\beta}{2\beta+d}}.$$

This completes the proof. ■

Notice that the rates obtained in Theorem 4.1, can be obtained in the full information case, where the operator observes the whole i.i.d sequence $(X_i, Y_i^{(1)}, Y_i^{(2)}), i = 1, \dots, n$, even before the first round. Indeed, such bounds have been obtained by Audibert and Tsybakov (2007) in the classification setup, i.e., when the rewards are Bernoulli random variables. However, we state a different technique, tailored for bandit policies in a partial information setup. While the final result is the same, we believe that it sheds light on the technicalities encountered in proving such a lower bound.

References

- Audibert, J.-Y., & Tsybakov, A. B. (2007). Fast learning rates for plug-in classifiers. *Ann. Statist.*, 35, 608–633.
- Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47, 235–256.
- Boucheron, S., Bousquet, O., & Lugosi, G. (2005). Theory of classification: a survey of some recent advances. *ESAIM Probab. Stat.*, 9, 323–375 (electronic).
- Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge: Cambridge University Press.
- Goldenshluger, A., & Zeevi, A. (2009). Woodroofe’s one-armed bandit problem revisited. *Ann. Appl. Probab.*, 19, 1603–1633.
- Hazan, E., & Megiddo, N. (2007). Online learning with prior knowledge. In *Learning theory*, vol. 4539 of *Lecture Notes in Comput. Sci.*, 499–513. Berlin: Springer.
- Kakade, S., Shalev-Shwartz, S., & Tewari, A. (2008). Efficient bandit algorithms for online multiclass prediction. *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)* (pp. 440–447). Helsinki, Finland: Omnipress.
- Lai, T. L., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Adv. in Appl. Math.*, 6, 4–22.

- Langford, J., & Zhang, T. (2008). The epoch-greedy algorithm for multi-armed bandits with side information. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*, 817–824. Cambridge, MA: MIT Press.
- Lu, T., Pál, D., & Pál, M. (2009). *Showing relevant ads via context multi-armed bandits* (Technical Report).
- Mammen, E., & Tsybakov, A. B. (1999). Smooth discrimination analysis. *Ann. Statist.*, 27, 1808–1829.
- Rigollet, P., & Vert, R. (2009). Fast rates for plug-in estimators of density level sets. *Bernoulli*, 15, 1154–1178.
- Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58, 527–535.
- Slivkins, A. (2009). Contextual bandits with similarity information. *Arxiv preprint arXiv:0907.3986*.
- Tsybakov, A. B. (2004). Optimal aggregation of classifiers in statistical learning. *Ann. Statist.*, 32, 135–166.
- Tsybakov, A. B. (2009). *Introduction to nonparametric estimation*. Springer Publishing Company, Incorporated.
- Wang, C.-C., Kulkarni, S., & Poor, H. (2005). Bandit problems with side observations. *Automatic Control, IEEE Transactions on*, 50, 338–355.
- Woodroffe, M. (1979). A one-armed bandit problem with a concomitant variable. *J. Amer. Statist. Assoc.*, 74, 799–806.
- Yang, Y., & Zhu, D. (2002). Randomized allocation with nonparametric estimation for a multi-armed bandit problem with covariates. *Ann. Statist.*, 30, 100–121.

An Asymptotically Optimal Bandit Algorithm for Bounded Support Models

Junya Honda and Akimichi Takemura
The University of Tokyo, Japan.
{honda, takemura}@stat.t.u-tokyo.ac.jp

Abstract

Multiarmed bandit problem is a typical example of a dilemma between exploration and exploitation in reinforcement learning. This problem is expressed as a model of a gambler playing a slot machine with multiple arms. We study stochastic bandit problem where each arm has a reward distribution supported in a known bounded interval, e.g. $[0, 1]$. In this model, Auer et al. (2002) proposed practical policies called UCB and derived finite-time regret of UCB policies. However, policies achieving the asymptotic bound given by Burnetas and Katehakis (1996) have been unknown for the model. We propose Deterministic Minimum Empirical Divergence (DMED) policy and prove that DMED achieves the asymptotic bound. Furthermore, the index used in DMED for choosing an arm can be computed easily by a convex optimization technique. Although we do not derive a finite-time regret, we confirm by simulations that DMED achieves a regret close to the asymptotic bound in finite time.

1 Introduction

The multiarmed bandit problem is a problem based on an analogy with a gambler playing a slot machine with more than one arm or lever. The objective of the gambler is to maximize the collected sum of rewards by choosing an arm to pull for each round. There is a dilemma between exploration and exploitation.

We consider a K -armed stochastic bandit problem. There are K arms Π_1, \dots, Π_K and each Π_j has a probability distribution F_j with the expected value μ_j . The gambler chooses an arm to pull based on a policy and receives a reward according to F_j independently in each round. If the expected values of the arms are known, it is optimal to always pull the arm with the maximum expected value $\mu^* = \max_j \mu_j$. There have been many studies for this problem (Agrawal, 1995; Even-Dar et al., 2002; Meuleau & Bourguine, 1999; Strens, 2000; Vermorel & Mohri, 2005; Yakowitz & Lowe, 1991). There are also many extensions for the problem, such as non-stationary distributions (Gittins, 1989; Ishikida & Varaiya, 1994) and non-stochastic bandit (Auer et al., 2003).

Lai and Robbins (1985) constructed a theoretical framework for determining optimal policies. Burnetas and Katehakis (1996) extended their result to multiparameter or non-parametric models which is relevant to our setting. Consider a model \mathcal{F} , that is, a generic family of distributions. The player knows \mathcal{F} and that each F_j is an element of \mathcal{F} . Let $T_j(n)$ denote the number of times that Π_j has been pulled over the first n rounds. Π_i is called suboptimal if $\mu_i < \mu^*$. A policy is *consistent* on model \mathcal{F} if $E[T_i(n)] = o(n^a)$ for all suboptimal arms Π_i and all $a > 0$.

Burnetas and Katehakis proved the following lower bound for any suboptimal Π_i under consistent policy:

$$T_i(n) \geq \left(\frac{1}{\inf_{G \in \mathcal{F}: E(G) > \mu^*} D(F_i || G)} + o(1) \right) \log n \quad (1)$$

with probability tending to one, where $E(G)$ is the expected value of distribution G and $D(\cdot || \cdot)$ denotes the Kullback-Leibler divergence. Under mild regularity conditions on \mathcal{F} ,

$$\inf_{G \in \mathcal{F}: E(G) > \mu} D(F || G) = \inf_{G \in \mathcal{F}: E(G) \geq \mu} D(F || G)$$

and we write

$$D_{\min}(F, \mu) = \inf_{G \in \mathcal{F}: E(G) \geq \mu} D(F || G).$$

A policy is *asymptotically optimal* if the expected value of $T_j(n)$ achieves the right-hand side of (1) as $n \rightarrow \infty$. Lai and Robbins (1985) and Burnetas and Katehakis (1996) also proposed policies based on the notion of *upper confidence bound* and proved their optimality for some specific models. Furthermore, Auer et al. (2002) proposed some practical policies called UCB for other models. UCB policies estimate the expectation of each arm in a similar way to upper confidence bound. Note that this framework of the simple K -armed stochastic bandit problem is also important for applications. It is because efficient policies for this simple problem are also bases of some extended framework of bandit problems, such as Kleinberg (2005) for uncountable arms or Kleinberg et al. (2008) for the case that some arms can not be chosen at some rounds.

Now consider our model \mathcal{A} , the family of distributions on a known interval, e.g. $[0, 1]$. This model \mathcal{A} represents one of the most basic nonparametric bandit situations. In this model, UCB policies are popular for their simple form and fine performance. However, although the performance of UCB policies is assured theoretically by a non-asymptotic form, their coefficients of the logarithmic term only depend on the expectations and the variances of arms and do not depend on the distributions themselves. Therefore, these theoretical analyses do not necessarily achieve the bound (1).

In this paper we propose Deterministic Minimum Empirical Divergence (DMED) policy. We prove the asymptotic optimality of DMED for our model \mathcal{A} . Although we do not give a finite bound of DMED as opposed to UCB policies, we confirm by simulations that DMED achieves performance close to the asymptotic bound in finite time.

Our DMED policy is motivated by a Bayesian viewpoint for the problem (although we do not use a Bayesian framework for theoretical analyses). Consider the case $K = 2$ and assume that Π_1 seems to be the best and $n \approx T_1(n) \gg T_2(n)$ at the n -th round. In this case the maximum likelihood that Π_1 and Π_2 are the best are roughly 1 and $\exp(-T_2(n)D_{\min}(\hat{F}_2, \hat{\mu}^*))$, respectively, where \hat{F}_2 is the empirical distribution of rewards from Π_2 and $\hat{\mu}^*$ is the current best sample mean. Then, the posterior expectation of the regret is proportional to $T_2(n) \cdot 1 + n \cdot \exp(-T_2(n)D_{\min}(\hat{F}_2, \hat{\mu}^*))$. DMED tries to minimize this by balancing these two terms. Note that DMED requires a computation of $D_{\min}(\hat{F}_i, \hat{\mu}^*) = \inf_{G \in \mathcal{A}: \mathbb{E}(G) \geq \hat{\mu}^*} D(\hat{F}_i || G)$ at each round. As shown in Theorem 8 below, D_{\min} can be expressed as a univariate convex optimization problem and it can be computed efficiently.

This paper is organized as follows. In Section 2, we give definitions used throughout this paper and recall the asymptotic bound by Burnetas and Katehakis (1996). In Section 3, we propose DMED policy which achieves the asymptotic bound. In Section 4, we analyze $D_{\min}(F, \mu)$ as an optimal value function for a practical implementation and a proof of the optimality of DMED. In Section 5, we prove the asymptotic optimality of DMED by the results of Section 4. Some simulation results are shown in Section 6. We conclude the paper with some remarks in Section 7.

2 Preliminaries

In this section we introduce notation of this paper and present the asymptotic bound for a generic model, which is established by Burnetas and Katehakis (1996).

Let \mathcal{F} be a generic family of probability distributions on \mathbb{R} and let $F_j \in \mathcal{F}$ be the distribution of Π_j , $j = 1, \dots, K$. $P_F[\cdot]$ and $\mathbb{E}_F[\cdot]$ denote the probability and the expectation under $F \in \mathcal{F}$, respectively. When we write e.g. $P_F[X \in A]$ ($A \subset \mathbb{R}$) or $\mathbb{E}_F[\theta(X)]$ ($\theta(\cdot)$ is a function $\mathbb{R} \rightarrow \mathbb{R}$), X denotes a random variable with distribution F . We define $F(A) \equiv P_F[X \in A]$ and $\mathbb{E}(F) \equiv \mathbb{E}_F[X]$.

A set of probability distributions for K arms is denoted by $\mathbf{F} \equiv (F_1, \dots, F_K) \in \mathcal{F}^K \equiv \prod_{j=1}^K \mathcal{F}$. The joint probability and the expected value under \mathbf{F} are denoted by $P_{\mathbf{F}}[\cdot]$, $\mathbb{E}_{\mathbf{F}}[\cdot]$, respectively.

The expected value of Π_j is denoted by $\mu_j \equiv \mathbb{E}(F_j)$. We denote the optimal expected value by $\mu^* \equiv \max_j \mu_j$. Let J_n denote the arm chosen in the n -th round. Then

$$T_j(n) = \sum_{m=1}^n \mathbb{I}[J_m = j],$$

where $\mathbb{I}[\cdot]$ denotes the indicator function.

Let $\hat{F}_{j,t}$ and $\hat{\mu}_{j,t} \equiv \mathbb{E}(\hat{F}_{j,t})$ be the empirical distribution and the mean of the first t rewards from Π_j , respectively. Similarly, let $\hat{F}_j(n) \equiv \hat{F}_{j,T_j(n)}$ and $\hat{\mu}_j(n) \equiv \hat{\mu}_{j,T_j(n)}$ be the empirical distribution and the mean of Π_j after the first n rounds, respectively. $\hat{\mu}^*(n) \equiv \max_j \hat{\mu}_j(n)$ denotes the highest empirical mean after the first n rounds. We call Π_j a current best if $\hat{\mu}_j(n) = \hat{\mu}^*(n)$.

The joint probability of two events A and B under \mathbf{F} is written as $P_{\mathbf{F}}[A \cap B]$. For notational simplicity we often write, e.g., $P_{\mathbf{F}}[J_n = j \cap T_j(n) = t]$ instead of the more precise $P_{\mathbf{F}}[\{J_n = j\} \cap \{T_j(n) = t\}]$.

Finally we define an index for $F \in \mathcal{F}$ and $\mu \in \mathbb{R}$

$$D_{\text{inf}}(F, \mu, \mathcal{F}) \equiv \inf_{G \in \mathcal{F}: \mathbb{E}(G) > \mu} D(F || G)$$

where Kullback-Leibler divergence $D(F||G)$ is given by

$$D(F||G) \equiv \begin{cases} \mathbb{E}_F \left[\log \frac{dF}{dG} \right] & \frac{dF}{dG} \text{ exists,} \\ +\infty & \text{otherwise.} \end{cases}$$

D_{inf} represents how distinguishable F is from distributions having expectations larger than μ . If $\{G \in \mathcal{F} : \mathbb{E}(G) > \mu\}$ is empty, we define $D_{\text{inf}}(F, \mu, \mathcal{F}) = +\infty$.

Theorem 2 of Lai and Robbins (1985) gave a lower bound for $\mathbb{E}[T_i(n)]$ for any suboptimal Π_i when a consistent policy is adopted. However their result was hard to apply for multiparameter models and more general non-parametric models. Later Burnetas and Katehakis (1996) extended the bound to general non-parametric models. Their bound is given as follows.

Proposition 1 (Proposition 1 of Burnetas and Katehakis (1996)) *Fix a consistent policy and $F \in \mathcal{F}^K$. If $\mu_i < \mu^*$ and $0 < D_{\text{inf}}(F_i, \mu^*, \mathcal{F}) < \infty$, then for any $\epsilon > 0$*

$$\lim_{N \rightarrow \infty} P_{\mathbf{F}} \left[T_i(N) \geq \frac{(1 - \epsilon) \log N}{D_{\text{inf}}(F_i, \mu^*, \mathcal{F})} \right] = 1.$$

Consequently

$$\liminf_{N \rightarrow \infty} \frac{\mathbb{E}_{\mathbf{F}}[T_i(N)]}{\log N} \geq \frac{1}{D_{\text{inf}}(F_i, \mu^*, \mathcal{F})}. \quad (2)$$

3 An Asymptotically Optimal Policy

Let $\mathcal{A} \equiv \{G : \text{supp}(G) \subset [a, b]\}$ be the family of distributions with a bounded support, where $\text{supp}(G)$ is the support of distribution G and a, b are constants known to the player. We assume $a = 0, b = 1$ without loss of generality. We consider $\mathcal{A} = \{G : \text{supp}(G) \subset [0, 1]\}$ as a model \mathcal{F} for the rest of this paper.

When we adopt the model \mathcal{A} , it is convenient to use

$$D_{\text{min}}(F, \mu, \mathcal{A}) \equiv \inf_{G \in \mathcal{A}: \mathbb{E}(G) \geq \mu} D(F||G)$$

instead of $D_{\text{inf}}(F, \mu, \mathcal{A}) = \inf_{G \in \mathcal{A}: \mathbb{E}(G) > \mu} D(F||G)$.

Lemma 2 $D_{\text{min}}(F, \mu, \mathcal{A}) = D_{\text{inf}}(F, \mu, \mathcal{A})$ holds for all $F \in \mathcal{A}$ and $\mu < 0$.

Proof: $D_{\text{min}}(F, \mu, \mathcal{A}) \leq D_{\text{inf}}(F, \mu, \mathcal{A}) \leq D_{\text{min}}(F, \mu + \epsilon, \mathcal{A})$ holds for arbitrary $\epsilon > 0$ from the definitions of D_{min} and D_{inf} . $D_{\text{min}}(F, \mu, \mathcal{A}) = D_{\text{inf}}(F, \mu, \mathcal{A})$ follows by letting $\epsilon \downarrow 0$, since we will prove in Theorem 7 that $D_{\text{min}}(F, \mu, \mathcal{A})$ is continuous in $\mu < 0$. \blacksquare

We simply write $D_{\text{min}}(F, \mu) \equiv D_{\text{min}}(F, \mu, \mathcal{A})$ when the third argument is obvious from the context. We discuss properties of D_{min} in Section 4.

Now we introduce Deterministic Minimum Empirical Divergence (DMED) policy and show its asymptotic optimality. We named it ‘‘deterministic’’ because our initial proposal, MED in Honda and Takemura (2010), was a randomized policy.

In the following algorithm, some arms are pulled once in one loop. Through the loop, arms to be pulled in the next loop are chosen and added to a list (a set) of arms. L_C denotes the list of arms to be pulled in the current loop. L_N denotes the list of arms to be pulled in the next loop. $L_R \subset L_C$ denotes the list of remaining arms of L_C which have not yet been pulled in the current loop. Arms are listed in L_N according to the occurrence of the event $J'_n(j)$ given by

$$J'_n(j) \equiv \{T_j(n) D_{\text{min}}(\hat{F}_j(n), \hat{\mu}^*(n)) \leq \log n - \log T_j(n)\}. \quad (3)$$

[Deterministic Minimum Empirical Divergence Policy]

Initialization. $L_C, L_R := \{1, \dots, K\}, L_N := \emptyset$. Pull each arm once. $n := K$.

Loop.

1. For $i \in L_C$ in the ascending order,
 - 1.1. $n := n + 1$ and pull Π_i . $L_R := L_R \setminus \{i\}$.
 - 1.2. $L_N := L_N \cup \{j\}$ (without a duplicate) for all $j \notin L_R$ such that $J'_n(j)$ occurs.
2. $L_C, L_R := L_N$ and $L_N := \emptyset$.

As shown above, $|L_C|$ arms are played in one loop. At every round, Π_j is added to L_N if $j \notin L_R$ and $J'_n(j)$ occurs. Note that if Π_j is a current best for the n -th round then $J'_n(j)$ holds since $D_{\min}(\hat{F}_j(n), \hat{\mu}^*(n)) = 0$ for this case. Then L_C is always not empty.

We use only the following fact as a property of DMED policy for our proof of the optimality:

Fact 3 (i) For any n it holds that $\sum_{m=1}^n \mathbb{I}[J_m = j] \leq 2 + \sum_{m=1}^n \mathbb{I}[J'_m(j)]$.

(ii) If $J'_{n_0}(j)$ occurs for any n_0 then $T_j(n) \geq T_j(n_0) + 1$ for all $n \geq n_0 + K$.

(i) and (ii) holds from the following reasons: (i) if Π_j is pulled at round $m > 2K$ then there exists a corresponding $n_m < m$ such that $J'_{n_m}(j)$ occurs and j is listed newly in L_N . The constant 2 is the effect of the initialization phase. (ii) There exists only three cases when $J'_{n_0}(j)$ occurs at the n_0 -th round: (1) j is listed newly in L_N , (2) j is already listed in L_N , (3) j is listed in L_R . In each case Π_j is pulled at least once through $n_0 + 1, \dots, n_0 + K$ -th rounds and $T_j(n)$ is incremented.

Theorem 4 Fix $F \in \mathcal{A}^K$ for which there exists j such that $\mu_j = \mu^*$ and $\mu_i < \mu^*$ for all $i \neq j$. Under DMED policy, for any $i \neq j$ and $\epsilon > 0$ it holds that

$$\mathbb{E}_F[T_i(N)] \leq \frac{1 + \epsilon}{D_{\min}(F_i, \mu^*)} \log N + O(1)$$

where $O(1)$ denotes a constant dependent on ϵ and F but independent of N .

Note that we obtain

$$\limsup_{N \rightarrow \infty} \frac{\mathbb{E}_F[T_i(N)]}{\log N} \leq \frac{1}{D_{\min}(F_i, \mu^*)},$$

by dividing both sides by $\log N$, letting $N \rightarrow \infty$ and finally letting $\epsilon \downarrow 0$. In view of (2) we see that DMED policy is asymptotically optimal. We prove Theorem 4 in Section 5 by using results on D_{\min} described in Section 4.

Note that the same bound as Theorem 4 can be derived when we substitute $\log T_j(n)$ in the criterion $J'_n(j)$ with an arbitrary constant or 0. However, we adopt the above criterion because simulation results seem better than that of other criteria. Our criterion may be justified by the Bayesian interpretation given in Introduction.

4 Analyses on Minimum Divergence

$D_{\min}(F, \mu)$ is the essential quantity for our DMED policy. In this section we introduce a dual problem $D'_{\min}(F, \mu)$ for $D_{\min}(F, \mu)$, which is computable efficiently. The main goal of this section is to show $D_{\min} = D'_{\min}$ and the continuity of them in F, μ . We discuss differentiability and continuity of $D'_{\min}(F, \mu)$ as a function of F and μ in Section 4.2. We show $D_{\min} = D'_{\min}$ in Section 4.3 by using the results of preceding subsections.

We now endow our model \mathcal{A} with a distance to define the continuities of $D_{\min}(F, \mu)$ and $D'_{\min}(F, \mu)$ in $F \in \mathcal{A}$ and closedness of a subset of \mathcal{A} . We adopt Lévy distance

$$L(F, G) \equiv \inf\{h > 0 : F((-\infty, x - h]) - h \leq G((-\infty, x]) \leq F((-\infty, x + h]) + h \text{ for all } x\}$$

for the distance between two distributions. Note that the convergence of the Lévy distance $L(F, F_n) \rightarrow 0$ is equivalent to the weak convergence of $\{F_n\}$ to distribution F and we write $F_n \rightarrow F$ in this sense (see, e.g., Lamperti (1996) for detail).

4.1 A Dual Problem

For $\mu < 0$, define

$$\begin{aligned} H(\nu, F, \mu) &\equiv \mathbb{E}_F[\log(1 - (X - \mu)\nu)] \\ H'(\nu, F, \mu) &\equiv \frac{\partial H(\nu, F, \mu)}{\partial \nu} = -\mathbb{E}_F\left[\frac{X - \mu}{1 - (X - \mu)\nu}\right] \\ H''(\nu, F, \mu) &\equiv \frac{\partial^2 H(\nu, F, \mu)}{\partial \nu^2} = -\mathbb{E}_F\left[\frac{(X - \mu)^2}{(1 - (X - \mu)\nu)^2}\right] \end{aligned} \quad (4)$$

and

$$D'_{\min}(F, \mu) \equiv \max_{0 \leq \nu \leq \frac{1}{1-\mu}} H(\nu, F, \mu). \quad (5)$$

D'_{\min} corresponds to the Lagrangian dual problem for D_{\min} . D'_{\min} is a univariate convex optimization problem and it can be computed efficiently by iterative methods such as Newton's method (see, e.g., Boyd and Vandenberghe (2004) for general methods of convex programming).

We write $H(\nu)$, $H'(\nu)$, $H''(\nu)$ when we regard them as a function of ν and when other arguments are obvious from the context. Note that $H(\nu)$ is concave and strictly concave except for the degenerate case $F(\{\mu\}) = 1$ from (4). Now we define an optimal solution for (5) as

$$\nu^*(F, \mu) \equiv \operatorname{argmax}_{0 \leq \nu \leq \frac{1}{1-\mu}} H(\nu, F, \mu).$$

Note that $\nu^*(F, \mu)$ is unique except for the case $F(\{\mu\}) = 1$ from the strict concavity of $H(\nu, F, \mu)$ in ν . For the case $F(\{\mu\}) = 1$, $D'_{\min}(F, \mu) = H(\nu, F, \mu)$ holds for all $\nu \in [0, (1-\mu)^{-1}]$ and we define $\nu^*(F, \mu) \equiv (1-\mu)^{-1}$. We write $\nu^*(F)$ or more simply ν^* when other arguments are obvious from the context.

The following theorem is used through proofs in Section 4 and 5.

Theorem 5 Define $\mathbb{E}_F[(1-\mu)/(1-X)] = \infty$ for the case $F(\{1\}) > 0$. If $\mu \leq \mathbb{E}(F)$ then $D'_{\min}(F, \mu) = 0$. If $\mathbb{E}(F) \leq \mu$ and $\mathbb{E}_F[(1-\mu)/(1-X)] \leq 1$ then $\nu^* = (1-\mu)^{-1}$ and (5) is simply written as

$$D'_{\min}(F, \mu) = H\left(\frac{1}{1-\mu}\right) = \mathbb{E}_F\left[\log \frac{1-X}{1-\mu}\right].$$

If $\mathbb{E}(F) \leq \mu$ and $\mathbb{E}_F[(1-\mu)/(1-X)] \geq 1$ then ν^* satisfies $H'(\nu^*) = 0$ and

$$\mathbb{E}_F\left[\frac{1}{1-(X-\mu)\nu^*}\right] = 1, \quad \mathbb{E}_F\left[\frac{X}{1-(X-\mu)\nu^*}\right] = \mu. \quad (6)$$

Proof: $D'_{\min}(F, \mu) = 0$ for $\mu \leq \mathbb{E}(F)$ follows from $H(0) = 0$, $H'(0) = \mu - \mathbb{E}(F) \leq 0$ and the concavity of $H(\nu)$. $\nu^* = (1-\mu)^{-1}$ for the case $\mathbb{E}_F[(1-\mu)/(1-X)] \leq 1$ follows from $H'((1-\mu)^{-1}) = (1-\mu)(1 - \mathbb{E}_F[(1-\mu)/(1-X)]) \geq 0$ and the concavity of $H(\nu)$.

Finally we consider the case $\mathbb{E}(F) \leq \mu$ and $\mathbb{E}_F[(1-\mu)/(1-X)] \geq 1$. For this case $H'(0) = \mu - \mathbb{E}(F) \geq 0$ and $H'((1-\mu)^{-1}) = (1-\mu)(1 - \mathbb{E}_F[(1-\mu)/(1-X)]) \leq 0$. Therefore $H'(\nu^*) = 0$ hold from the concavity of $H(\nu)$. (6) follow from

$$\mathbb{E}_F\left[\frac{1}{1-(X-\mu)\nu^*}\right] = \mathbb{E}_F\left[\frac{1-(X-\mu)\nu^*}{1-(X-\mu)\nu^*}\right] + \nu^* \mathbb{E}_F\left[\frac{X-\mu}{1-(X-\mu)\nu^*}\right] = 1 - \nu^* H'(\nu^*) = 1$$

and

$$\mathbb{E}_F\left[\frac{X}{1-(X-\mu)\nu^*}\right] = \mathbb{E}_F\left[\frac{X-\mu}{1-(X-\mu)\nu^*}\right] + \mu \mathbb{E}_F\left[\frac{1}{1-(X-\mu)\nu^*}\right] = -H'(\nu^*) + \mu = \mu. \quad \blacksquare$$

4.2 Continuity and Differentiability of the Dual Problem

In this subsection we discuss the differentiability and the continuity of $D'_{\min}(F, \mu)$ in F and μ . We will show $D_{\min} = D'_{\min}$ in the next subsection and the result for D'_{\min} in this subsection also holds for D_{\min} .

Theorem 6 $D'_{\min}(F, \mu)$ is differentiable in $\mu \in (\mathbb{E}(F), 1)$ for any $F \in \mathcal{A}$ with

$$\frac{\partial}{\partial \mu} D'_{\min}(F, \mu) = \nu^*$$

We omit the proof but it can be proved by Corollary 3.4.3 of Fiacco (1983), which gives the differentiability of an optimal value function with parameters.

Theorem 7 $D'_{\min}(F, \mu)$ is continuous in (i) $\mu < 1$ and (ii) $F \in \mathcal{A}$.

Proof: (i) The continuity in μ is obvious in the interval $(\mathbb{E}(F), 1)$ from the differentiability in Theorem 6. The continuity in $\mu < \mathbb{E}(F)$ is also obvious since $D'_{\min}(F, \mu) = 0$ holds for all $\mu < \mathbb{E}(F)$. Finally we consider the continuity at $\mu = \mathbb{E}(F)$. From (5) and the concavity of $H(\nu)$, it holds that

$$H(0) \leq D'_{\min}(F, \mu) \leq \max\{H(0), H(0) + H'(0)\frac{1}{1-\mu}\}$$

or equivalently

$$0 \leq D'_{\min}(F, \mu) \leq \max\left\{0, \frac{\mu - \mathbb{E}(F)}{1-\mu}\right\}.$$

Then $\lim_{\mu \rightarrow \mathbb{E}(F)} D'_{\min}(F, \mu) = D'_{\min}(F, \mathbb{E}(F)) = 0$ is obtained by letting $\mu \rightarrow \mathbb{E}(F)$.

(ii) We consider the lower semicontinuity and the upper semicontinuity separately.

First we show the lower semicontinuity. Fix an arbitrary $\epsilon > 0$. From (5) and the continuity of $H(\nu)$, there exists $\nu_0 \in [0, (1 - \mu)^{-1}]$ such that $E_F[\log(1 - (X - \mu)\nu_0)] \geq D'_{\min}(F, \mu) - \epsilon$. Then we obtain

$$\begin{aligned} \liminf_{F' \rightarrow F} D'_{\min}(F', \mu) &\geq \liminf_{F' \rightarrow F} E_{F'}[\log(1 - (X - \mu)\nu_0)] \\ &= E_F[\log(1 - (X - \mu)\nu_0)] \\ &\geq D'_{\min}(F, \mu) - \epsilon. \end{aligned} \quad (7)$$

Note that $\log(1 - (x - \mu)\nu_0)$ is continuous and bounded in $x \in [0, 1]$ and (7) follows from the definition of weak convergence. The lower semicontinuity holds since ϵ is arbitrary.

Next we prove the upper semicontinuity. First we consider the case $E(F) > \mu$. In this case, $E(F') > \mu$ holds for all F' sufficiently close to F . Then $D'_{\min}(F, \mu) = D'_{\min}(F', \mu) = 0$ holds and the upper semicontinuity is obtained.

Next we consider the case $E_F[(1 - \mu)/(1 - X)] > 1$ and $E(F) \leq \mu$. Since $\nu^*(F) < (1 - \mu)^{-1}$ in this case, we obtain

$$\begin{aligned} \limsup_{F' \rightarrow F} D'_{\min}(F', \mu) &\leq \limsup_{F' \rightarrow F} \left(H(\nu^*(F), F', \mu) + \frac{1}{1 - \mu} |H'(\nu^*(F), F', \mu)| \right) \quad (\text{by the concavity of } H(\nu)) \\ &= H(\nu^*(F), F, \mu) + \frac{1}{1 - \mu} |H'(\nu^*(F), F, \mu)| \quad (\text{by the definition of weak convergence}) \\ &= D'_{\min}(F, \mu) \end{aligned}$$

and the upper semicontinuity is proved for this case.

For the case $E_F[(1 - \mu)/(1 - X)] \leq 1$, we omit the proof for lack of space. ■

The proof of the upper semicontinuity is a little complicated for the last case $E_F[(1 - \mu)/(1 - X)] \leq 1$. It is because $\nu^* = (1 - \mu)^{-1}$ holds for the case and $H(\nu, F, \mu)$ is difficult to analyze at $\nu = (1 - \mu)^{-1}$. In fact, in every neighborhood of F , there exists $G \in \mathcal{A}$ such that $H((1 - \mu)^{-1}, G, \mu) = H'((1 - \mu)^{-1}, G, \mu) = -\infty$. The upper semicontinuity can be proved by using the definition of the Lévy distance explicitly.

4.3 Equality of Minimum Divergence with the Dual Problem

In this subsection we prove $D_{\min} = D'_{\min}$ in Theorem 8. Therefore we can compute D_{\min} efficiently by solving the univariate convex optimization in D'_{\min} . Furthermore, the differentiability and the continuity in Theorem 6 and 7 also hold for D_{\min} .

Theorem 8 $D_{\min}(F, \mu) = D'_{\min}(F, \mu)$ holds for all $F \in \mathcal{A}$ and $\mu < 1$.

To prove this theorem, we additionally define \mathcal{A}_f and $\mathcal{A}_f(F)$, families of distributions with finite supports by

$$\begin{aligned} \mathcal{A}_f &\equiv \{G \in \mathcal{A} : |\text{supp}(G)| < \infty\}, \\ \mathcal{A}_f(F) &\equiv \{G \in \mathcal{A}_f : \text{supp}(G) \subset \text{supp}'(F)\} \quad (F \in \mathcal{A}_f) \end{aligned}$$

where $\text{supp}'(F) \equiv \{1\} \cup \text{supp}(F)$. Note that $\mathcal{A}_f(F) \subset \mathcal{A}_f \subset \mathcal{A}$ for all $F \in \mathcal{A}_f$.

Lemma 9 $D_{\min}(F, \mu, \mathcal{A}) = D_{\min}(F, \mu, \mathcal{A}_f(F))$ holds for all $F \in \mathcal{A}_f$.

We omit the proof but it can be proved by the following fact: if $G(A) \geq G'(A)$ for all $A \subset \text{supp}(F)$ then $D(F||G) \leq D(F||G')$.

Before proving $D_{\min}(F, \mu) = D'_{\min}(F, \mu)$ for general $F \in \mathcal{A}$, we show the equality for $F \in \mathcal{A}_f$ and $E(F) < \mu < 1$ by the technique of Lagrange multipliers.

Lemma 10 If $E(F) < \mu < 1$ and $F \in \mathcal{A}_f$ then $D_{\min}(F, \mu) = D'_{\min}(F, \mu)$ holds.

Proof (Sketch): Let $M \equiv |\text{supp}'(F)|$ and denote the finite symbols in $\text{supp}'(F)$ by x_1, \dots, x_M , i.e. $\{1\} \cup \text{supp}(F) = \{x_1, \dots, x_M\}$. We assume $x_1 = 1$ and $x_i < 1$ for $i > 1$ without loss of generality and write $f_i \equiv F(\{x_i\})$. $D_{\min}(F, \mu)$ is expressed as the following parametric convex optimization problem for $G = (g_1, \dots, g_M)$ from Lemma 9:

$$\text{minimize : } \sum_{i=1}^M f_i \log \frac{f_i}{g_i}, \quad \text{subject to : } g_i \geq 0, \forall i, \quad \sum_{i=1}^M x_i g_i \geq \mu, \quad \sum_{i=1}^M g_i = 1.$$

It is checked by the technique of Lagrange multipliers (see e.g. Section 28 of Rockafellar (1970)) that the optimal solution is

$$g_i^* = \begin{cases} \frac{1-\mu}{1-x_i} f_i & i \neq 1 \\ 1 - \sum_{i=2}^M \frac{1-\mu}{1-x_i} f_i & i = 1, \end{cases}$$

for the case $E_F[(1-\mu)/(1-X)] \leq 1$ and

$$g_i^* = \begin{cases} 0 & i = 1 \text{ and } f_1 = 0, \\ \frac{f_i}{1-(x_i-\mu)\nu^*} & \text{otherwise} \end{cases}$$

for the case $E_F[(1-\mu)/(1-X)] \geq 1$ from (6). The lemma is proved immediately from these expressions of $\{g_i^*\}$. \blacksquare

Proof of Theorem 8: It is easy to check that $D_{\min}(F, \mu) = D'_{\min}(F, \mu) = 0$ for $\mu \leq E(F)$. Hence we consider the case $E(F) < \mu < 1$.

First we prove $D'_{\min}(F, \mu) \geq D_{\min}(F, \mu)$. Define a measure G^* on $[0, 1]$ as

$$G^*(A) \equiv \begin{cases} \int_A \frac{1-\mu}{1-x} dF + (1 - E_F[(1-\mu)/(1-X)])\mathbb{I}[0 \in A] & E_F[(1-\mu)/(1-X)] \leq 1 \\ \int_A \frac{dF}{1-(x-\mu)\nu^*} & E_F[(1-\mu)/(1-X)] > 1. \end{cases}$$

To prove $D'_{\min}(F, \mu) \geq D_{\min}(F, \mu)$, it is sufficient to show that G^* is a probability measure with $E(G^*) \geq \mu$ since $D(F||G^*) = D'_{\min}(F, \mu)$. It is checked easily for the case $E_F[(1-\mu)/(1-X)] \leq 1$. For the case $E_F[(1-\mu)/(1-X)] > 1$, it is checked from (6).

Next we prove $D_{\min}(F, \mu) \geq D'_{\min}(F, \mu)$. Take an arbitrary $G \in \mathcal{A}$ satisfying $E(G) \geq \mu$. Consider a finite partition $\{U_i\}_{i=0, \dots, n}$ of $[0, 1]$:

$$U_i \equiv \begin{cases} \{0\} & i = 0 \\ \left(\frac{i-1}{n}, \frac{i}{n}\right] & i = 1, \dots, n \end{cases}$$

and define $F^n, G^n \in \mathcal{A}_f$ as

$$F^n \left(\left\{\frac{i}{n}\right\}\right) \equiv F(U_i), \quad G^n \left(\left\{\frac{i}{n}\right\}\right) \equiv G(U_i).$$

Then we have

$$\begin{aligned} D(F||G) &\geq D(F^n||G^n) && \text{(by Theorem 2.4.2 of Pinsker (1964))} \\ &\geq D_{\min}(F^n, \mu) && \text{(by } E(G^n) \geq E(G) \geq \mu) \\ &= D'_{\min}(F^n, \mu) && \text{(by Lemma 10)} \end{aligned} \tag{8}$$

Note that $L(F^n, F) \leq 1/n$ then $F^n \rightarrow F$ as $n \rightarrow \infty$. Therefore it holds for any $\epsilon > 0$ that

$$D'_{\min}(F^n, \mu) \geq D'_{\min}(F, \mu) - \epsilon \tag{9}$$

for sufficiently large n from the lower semicontinuity of $D'_{\min}(F, \mu)$ in F .

From (8) and (9) we obtain for all G satisfying $E(G) \geq \mu$ that

$$D(F||G) \geq D'_{\min}(F, \mu) - \epsilon$$

and

$$D_{\min}(F, \mu) \geq D'_{\min}(F, \mu) - \epsilon.$$

$D_{\min}(F, \mu) \geq D'_{\min}(F, \mu)$ follows since $\epsilon > 0$ is arbitrary. \blacksquare

5 A Proof of Theorem 4

Before proving Theorem 4, we show Lemmas 11–14 on properties of D_{\min} and ν^* .

Lemma 11 $D_{\min}(F, \mu)$ is monotonically increasing in μ .

This lemma follows immediately from the definition $D_{\min}(F, \mu) = \min_{G \in \mathcal{A}: E(G) \geq \mu} D(F||G)$. We use this monotonicity in the proof of Theorem 4 implicitly.

Lemma 12 If $E(F) < \mu$ then $\nu^* = \nu^*(F, \mu)$ satisfies

$$\nu^* \geq \frac{\mu - E(F)}{\mu(1 - \mu)}.$$

Proof: This lemma is easily checked for the case $\mathbb{E}_F[(1 - \mu)/(1 - X)] \leq 1$ from $\nu^* = (1 - \mu)^{-1}$ and we consider the case $\mathbb{E}_F[(1 - \mu)/(1 - X)] \geq 1$. Define

$$w(x, \nu) \equiv \frac{x - \mu}{1 - (x - \mu)\nu}.$$

For any fixed $\nu \in [0, (1 - \mu)^{-1}]$, $w(x, \nu)$ is convex in $x \in [0, 1]$. Therefore

$$\begin{aligned} H'(\nu) &= -\mathbb{E}_F[w(X, \nu)] \\ &\geq -\mathbb{E}_F[(1 - X)w(0, \nu) + Xw(1, \nu)] \\ &= (\mathbb{E}(F) - 1)w(0, \nu) - \mathbb{E}(F)w(1, \nu). \end{aligned} \quad (10)$$

The right-hand side of (10) is 0 for $\nu = (\mu - \mathbb{E}(F))/(\mu(1 - \mu))$ and therefore we obtain

$$H' \left(\frac{\mu - \mathbb{E}(F)}{\mu(1 - \mu)} \right) \geq 0.$$

The lemma is proved since $H'(\nu^*) = 0$ holds and H' is monotonically decreasing. ■

Lemma 13 Fix arbitrary $\mu, \mu' \in (0, 1)$ satisfying $\mu' < \mu$. Then there exists $C(\mu, \mu') > 0$ such that

$$D_{\min}(F, \mu) - D_{\min}(F, \mu') \geq C(\mu, \mu').$$

for all $F \in \mathcal{A}$ satisfying $\mathbb{E}(F) \leq \mu'$.

Proof: Since $D_{\min}(F, \mu)$ is differentiable in $\mu \in (\mathbb{E}(F), 1)$ and continuous in $\mu < 1$, we have

$$\begin{aligned} D_{\min}(F, \mu) - D_{\min}(F, \mu') &= \lim_{t \downarrow \mu'} \int_t^\mu \frac{\partial}{\partial u} D_{\min}(F, u) du \\ &\geq \lim_{t \downarrow \mu'} \int_t^\mu \frac{u - \mu'}{u(1 - u)} du \quad (\text{by Theorem 5 and Lemma 12}) \\ &\geq \frac{(\mu - \mu')^2}{2\mu(1 - \mu')} \quad (=: C(\mu, \mu')). \end{aligned} \quad \blacksquare$$

Lemma 14 $\sup_{G \in \mathcal{A}} D_{\min}(G, \mu) \leq -\log(1 - \mu) < +\infty$ for all $0 \leq \mu < 1$.

Proof: By applying Jensen's inequality for

$$D_{\min}(G, \mu) = \max_{0 \leq \nu \leq \frac{1}{1 - \mu}} \mathbb{E}_G[\log(1 - (X - \mu)\nu)],$$

we obtain

$$\begin{aligned} \sup_{G \in \mathcal{A}} D_{\min}(G, \mu) &\leq \sup_{G \in \mathcal{A}} \max_{0 \leq \nu \leq \frac{1}{1 - \mu}} \log(1 - (\mathbb{E}(G) - \mu)\nu) \\ &= \sup_{G \in \mathcal{A}} \max \left\{ 0, \log \frac{1 - \mathbb{E}(G)}{1 - \mu} \right\} \\ &= -\log(1 - \mu). \end{aligned} \quad \blacksquare$$

Proof of Theorem 4: We assume $j = 1$ and $\mu_2 = \max_{k \neq 1} \mu_k$ without loss of generality. Then $\mu_1 = \mu^*$ and $\mu_k \leq \mu_2$ for $k = 2, \dots, K$.

Note that $\mu_1 = 1$ is a trivial case and we assume $\mu_1 < 1$ in the following. For the case $\mu_1 = 1$, $F_1(\{1\}) = 1$ and $\mu^*(n)$ is always equal to 0. Therefore $J'_i(n)$ never occurs for sufficiently large n , because $D_{\min}(\hat{F}_i(n), 1) = +\infty$ always holds except for the case $\hat{F}_i(n) = F_1$.

We obtain from Fact 3 (i) that

$$T_i(N) = \sum_{n=1}^N \mathbb{I}[J_n = i] \leq 2 + \sum_{n=1}^N \mathbb{I}[J'_n(i)].$$

Now we define events A_n and B_n as

$$\begin{aligned} A_n &\equiv \{\hat{\mu}_1(n) \geq \mu_1 - \delta\}, \\ B_n &\equiv \{\hat{\mu}^*(n) \leq \mu_2 + \delta\} = \bigcap_{k=1}^K \{\hat{\mu}_k(n) \leq \mu_2 + \delta\}. \\ C_n &= \bigcup_{k=1}^K \{\hat{\mu}^*(n) = \hat{\mu}_k(n) \cap |\hat{\mu}_k(n) - \mu_k| \geq \delta\} \end{aligned}$$

where $\delta > 0$ is a constant satisfying $\mu_2 < \mu_1 - \delta$ and set sufficiently small in an evaluation on A_n . It is easily checked that $\{A_n^c \cap B_n^c\} \subset C_n$. Therefore $\mathbb{E}_{\mathbf{F}}[T_i(N)]$ is bounded as

$$\mathbb{E}_{\mathbf{F}}[T_i(N)] \leq 2 + \mathbb{E}_{\mathbf{F}} \left[\sum_{n=1}^N \mathbb{I}[J'_n(i) \cap A_n] \right] + \mathbb{E}_{\mathbf{F}} \left[\sum_{n=1}^N \mathbb{I}[B_n] \right] + \mathbb{E}_{\mathbf{F}} \left[\sum_{n=1}^N \mathbb{I}[C_n] \right]. \quad (11)$$

In the following Lemmas 15–17 we bound the right-hand side of (11) in this order and they prove the theorem. \blacksquare

Lemma 15 *For all $\epsilon > 0$ it holds that*

$$\mathbb{E}_{\mathbf{F}} \left[\sum_{n=1}^N \mathbb{I}[J'_n(i) \cap A_n] \right] \leq \frac{1 + \epsilon}{D_{\min}(F_i, \mu_1)} \log N + O(1).$$

Lemma 16

$$\mathbb{E}_{\mathbf{F}} \left[\sum_{n=1}^N \mathbb{I}[B_n] \right] = O(1).$$

Lemma 17

$$\mathbb{E}_{\mathbf{F}} \left[\sum_{n=1}^N \mathbb{I}[C_n] \right] = O(1).$$

Before proving these lemmas, we give intuitive interpretations for these terms.

$\sum_{n=1}^N \mathbb{I}[J'_n(i) \cap A_n]$ is the main term of $T_i(N)$. Roughly speaking, in DMED policy, Π_i is pulled and $T_i(n)$ is incremented until $T_i(n) D_{\min}(\hat{F}_i(n), \hat{\mu}^*(n))$ and $\log n - \log T_i(n) (\approx \log n)$ in (3) balance. Consider the following two cases on the event A_n :

- (1) If A_n happens and \hat{F}_i is sufficiently close to F_i , then $D_{\min}(\hat{F}_i, \hat{\mu}^*) \gtrsim D_{\min}(F_i, \mu^*)$ holds and the above two terms balance when $T_i(n) \lesssim \log n / D_{\min}(F_i, \mu^*)$, which is exactly the asymptotic bound to be achieved.
- (2) If A_n and $D_{\min}(\hat{F}_i, \hat{\mu}^*) < D_{\min}(F_i, \mu^*)$ happen, Π_i may be pulled more frequently than case (1). However, as Π_i is pulled, \hat{F}_i approaches F_i and $D_{\min}(\hat{F}_i, \hat{\mu}^*)$ approaches $D_{\min}(F_i, \mu^*)$. Then eventually $D_{\min}(\hat{F}_i, \hat{\mu}^*) < D_{\min}(F_i, \mu^*)$ does not hold and the effect of this event is not large.

The term involving B_n is essential for the consistency of DMED. If B_n occurs then $\hat{\mu}_1(n)$ is not yet close to μ_1 . It requires many rounds for Π_1 to be pulled since Π_1 may seem to be suboptimal in this event. Therefore B_n may happen for many n .

On the other hand when C_n occurs, empirical mean $\hat{\mu}_k(n)$ of current best Π_k is not close to the true expectation μ_k . Then Π_k is pulled more frequently and $\hat{\mu}_k(n)$ approaches μ_k . As a result, C_n happens only for a few n .

In the proofs of these three lemmas, we use Theorem 6.2.10 of Dembo and Zeitouni (1998) on the empirical distribution:

Proposition 18 (Sanov's Theorem) *For every closed set Γ of probability distributions (with respect to the Lévy distance),*

$$\limsup_{t \rightarrow \infty} \frac{1}{t} \log P_F[\hat{F}_t \in \Gamma] \leq - \inf_{G \in \Gamma} D(G||F).$$

where \hat{F}_t is the empirical distribution of t samples from F .

Proof of Lemma 15 (Sketch): By partitioning the event $J'_n(i)$ according to the value of $T_i(n)$, we obtain

$$\begin{aligned} & \sum_{n=1}^N \mathbb{I}[J'_n(i) \cap A_n] \\ &= \sum_{t=1}^N \mathbb{I} \left[\bigcup_{n=1}^N \left\{ \{t D_{\min}(\hat{F}_{i,t}, \hat{\mu}^*(n)) \leq \log n - \log t\} \cap A_n \cap T_i(n) = t \right\} \right] \end{aligned}$$

$$\begin{aligned}
&\leq \frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)} + \sum_{t=\frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)}}^N \mathbb{I} \left[\bigcup_{n=1}^N \left\{ \{tD_{\min}(\hat{F}_{i,t}, \hat{\mu}^*(n)) \leq \log n\} \cap A_n \cap T_i(n) = t \right\} \right] \\
&\leq \frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)} + \sum_{t=\frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)}}^N \mathbb{I} \left[\frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)} D_{\min}(\hat{F}_{i,t}, \mu_1 - \delta) \leq \log N \right] \\
&\hspace{25em} (\mu_1 - \delta \leq \hat{\mu}^*(n) \text{ on } A_n) \\
&= \frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)} + \sum_{t=\frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)}}^N \mathbb{I} \left[D_{\min}(\hat{F}_{i,t}, \mu_1 - \delta) \leq \frac{D_{\min}(F_i, \mu_1)}{1+\epsilon} \right]. \tag{12}
\end{aligned}$$

Define $\Gamma^\delta \equiv \{G \in \mathcal{A} : L(F_i, G) \geq \delta\}$. By applying Sanov's Theorem with $F := F_i$ and $\Gamma := \Gamma^\delta$, there exists C_1 such that

$$P_{F_i}[\hat{F}_{i,t} \in \Gamma^\delta] = O(\exp(-tC_1)). \tag{13}$$

Here we use the fact that for sufficiently small $\delta > 0$

$$\left\{ D_{\min}(\hat{F}_{i,t}, \mu_1 - \delta) \leq \frac{D_{\min}(F_i, \mu_1)}{1+\epsilon} \right\} \subset \{\hat{F}_{i,t} \in \Gamma^\delta\} \tag{14}$$

or equivalently $\{L(\hat{F}_{i,t}, F_i) < \delta\} \subset \{D_{\min}(\hat{F}_{i,t}, \mu_1 - \delta) > D_{\min}(F_i, \mu_1)/(1+\epsilon)\}$. This can be proved by the continuity in F and the differentiability in μ of $D_{\min}(F, \mu)$.

From (12), (13) and (14), we obtain

$$\begin{aligned}
\mathbb{E}_F \left[\sum_{n=1}^N \mathbb{I}[J_n(i) \cap A_n] \right] &\leq \frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)} + \sum_{t=\frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)}}^N O(\exp(-tC_1)) \\
&= \frac{(1+\epsilon)\log N}{D_{\min}(F_i, \mu_1)} + O(1). \quad \blacksquare
\end{aligned}$$

Proof of Lemma 16: Define $C_2 \equiv C(\mu_1, \mu_2 + \delta)/3$ and $Q \equiv \lceil \sup_{G \in \mathcal{A}} D_{\min}(G, \mu_2 + \delta)/C_2 \rceil$. $Q < +\infty$ holds from Lemma 14. Take a finite cover $\{S_q\}_{q=1,2,\dots,Q}$ of $\{G \in \mathcal{A} : \mathbb{E}(G) \leq \mu_2 + \delta\}$ as

$$S_q \equiv \{G \in \mathcal{A} : \mathbb{E}(G) \leq \mu_2 + \delta, (q-1)C_2 \leq D_{\min}(G, \mu_2 + \delta) \leq qC_2\}.$$

Since $D_{\min}(F, \mu)$ is continuous in F , each S_q is a closed set. By applying Sanov's Theorem with $F := F_1$ and $\Gamma := S_q$, there exists $t_q > 0$ such that for all $t > t_q$

$$\begin{aligned}
P_{F_1}[\hat{F}_{1,t} \in S_q] &\leq \exp\left(-t\left(\inf_{G \in S_q} D(G||F_1) - C_2\right)\right) \\
&\leq \exp\left(-t\left(\inf_{G \in S_q} D_{\min}(G, \mu_1) - C_2\right)\right) \\
&\leq \exp\left(-t\left(\inf_{G \in S_q} D_{\min}(G, \mu_2 + \delta) + C(\mu_1, \mu_2 + \delta) - C_2\right)\right) \text{ (by Lemma 13)} \\
&\leq \exp(-t(q+1)C_2). \quad \left(\text{by } \inf_{G \in S_q} D_{\min}(G, \mu_2 + \delta) \geq (q-1)C_2\right)
\end{aligned}$$

Therefore, by defining $t' \equiv \max_{q=1,\dots,Q} t_q$, it holds for all $t > t'$ that

$$P_{F_1}[\hat{F}_{1,t} \in S_q] \leq \exp(-t(q+1)C_2). \tag{15}$$

$\sum_{n=1}^N \mathbb{I}[B_n]$ is bounded as

$$\sum_{n=1}^{\infty} \mathbb{I}[B_n] \leq \sum_{q=1}^Q \sum_{t=1}^{\infty} \sum_{n=1}^{\infty} \mathbb{I}[B_n \cap T_1(n) = t \cap \hat{F}_{1,t} \in S_q] \tag{16}$$

since $\{\hat{F}_1(n) \in \bigcup_{q=1}^Q S_q\} = \{\hat{\mu}_1(n) \leq \mu_2 + \delta\} \supset B_n$. Now for each t and q we show that

$$\sum_{n=1}^{\infty} \mathbb{I}[B_n \cap T_1(n) = t \cap \hat{F}_{1,t} \in S_q] \leq t \exp(tqC_2) + K. \tag{17}$$

Assume that $\sum_{n=1}^{\infty} \mathbb{I}[B_n \cap T_1(n) = t \cap \hat{F}_{1,t} \in S_q] \geq t \exp(tqC_2)$. On this event, we can take an integer $m \geq t \exp(tqC_2)$ such that the events

$$B_m \cap T_1(m) = t \cap \hat{F}_{1,t} \in S_q \quad (18)$$

and

$$\sum_{n=1}^m \mathbb{I}[B_n \cap T_1(n) = t \cap \hat{F}_{1,t} \in S_q] = \lceil t \exp(tqC_2) \rceil \quad (19)$$

occur. For this m , it holds that

$$\begin{aligned} T_1(m)D_{\min}(\hat{F}_1(m), \hat{\mu}^*(m)) &\leq t \sup_{G \in S_q} D_{\min}(G, \mu_2 + \delta) \quad (\text{by (18)}) \\ &\leq tqC_2 \\ &\leq \log m - \log t. \quad (\text{by } m \geq t \exp(tqC_2)) \end{aligned}$$

Then $J'_m(1)$ holds and $T_1(n) \geq t + 1$ for all $n \geq m + K$ from Fact 3 (ii). Therefore we obtain (17) from

$$\begin{aligned} \sum_{n=1}^{\infty} \mathbb{I}[B_n \cap T_1(n) = t \cap \hat{F}_{1,t} \in S_q] &= \sum_{n=1}^{m+K-1} \mathbb{I}[B_n \cap T_1(n) = t \cap \hat{F}_{1,t} \in S_q] \\ &\leq \lceil t \exp(tqC_2) \rceil + K - 1 \quad (\text{by (19)}) \\ &\leq t \exp(tqC_2) + K. \end{aligned}$$

Now we obtain from (15), (16) and (17) that

$$\begin{aligned} \mathbb{E}_{\mathbf{F}} \left[\sum_{n=1}^N \mathbb{I}[B_n] \right] &\leq \sum_{q=1}^Q \sum_{t=1}^{\infty} P_{F_1}[\hat{F}_{1,t} \in S_q] (t \exp(tqC_2) + K) \\ &\leq \sum_{q=1}^Q \sum_{t=1}^{t'} (t \exp(tqC_2) + K) + \sum_{q=1}^Q \sum_{t=t'}^{\infty} \exp(-t(q+1)C_2) (t \exp(tqC_2) + K) \\ &\leq O(1) + Q \sum_{t=t'}^{\infty} (t \exp(-tC_2) + K \exp(-2tC_2)) = O(1). \quad \blacksquare \end{aligned}$$

Proof of Lemma 17: We obtain from the definition of C_n that

$$\begin{aligned} \sum_{n=1}^N \mathbb{I}[C_n] &\leq \sum_{k=1}^K \sum_{n=1}^{\infty} \mathbb{I}[\hat{\mu}^*(n) = \hat{\mu}_k(n) \cap |\hat{\mu}_k(n) - \mu_k| \geq \delta] \\ &\leq \sum_{k=1}^K \sum_{t=1}^{\infty} \sum_{n=1}^{\infty} \mathbb{I}[\hat{\mu}^*(n) = \hat{\mu}_{k,t} \cap |\hat{\mu}_{k,t} - \mu_k| \geq \delta \cap T_k(n) = t]. \end{aligned}$$

Suppose that $\hat{\mu}^*(n_0) = \hat{\mu}_{k,t} \cap T_k(n_0) = t$ occurs at n_0 -th round for the first time. Then Π_k is a current best at the n_0 -th round and $J'_{n_0}(k)$ holds. Therefore $T_k(n) \geq t + 1$ for all $n \geq n_0 + K$ from Fact 3 (ii). As a result, we obtain

$$\begin{aligned} \sum_{n=1}^{\infty} \mathbb{I}[\hat{\mu}^*(n) = \hat{\mu}_{k,t} \cap |\hat{\mu}_{k,t} - \mu_k| \geq \delta \cap T_k(n) = t] \\ = \sum_{n=n_0}^{n_0+K-1} \mathbb{I}[\hat{\mu}^*(n) = \hat{\mu}_{k,t} \cap |\hat{\mu}_{k,t} - \mu_k| \geq \delta \cap T_k(n) = t] \leq K \end{aligned}$$

and

$$\mathbb{E}_{\mathbf{F}} \left[\sum_{n=1}^N \mathbb{I}[C_n] \right] \leq K \sum_{k=1}^K \sum_{t=1}^{\infty} P_{F_k}[|\hat{\mu}_{k,t} - \mu_k| \geq \delta].$$

By applying Sanov's Theorem with $F := F_k$ and $\Gamma := \{G \in \mathcal{A} : |\mathbb{E}(G) - \mu_k| \geq \delta\}$, there exists $C_3 > 0$ such that $P_{F_k}[|\hat{\mu}_{k,t} - \mu_k| \geq \delta] = O(\exp(-tC_3))$. Now we obtain

$$\mathbb{E}_{\mathbf{F}} \left[\sum_{n=1}^N \mathbb{I}[C_n] \right] \leq K \sum_{k=1}^K \sum_{t=1}^{\infty} O(\exp(-tC_3)) = O(1). \quad \blacksquare$$

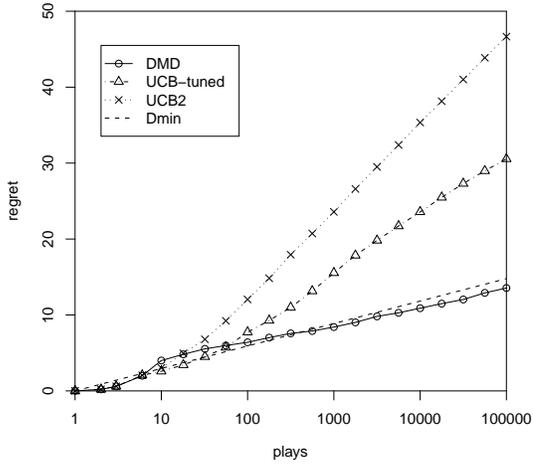


Figure 1: Experiment for beta distributions.

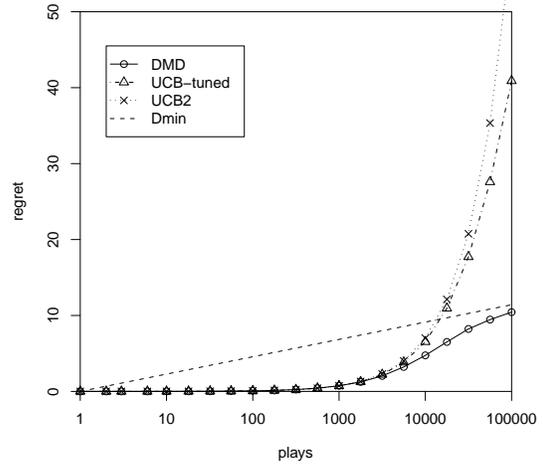


Figure 2: Experiment for distributions, which are hard to distinguish.

6 Experiments

In this section we give experimental results using UCB2, UCB-tuned (Auer et al., 2002) and DMED. In the implementation of DMED, $D_{\min}(\hat{F}_i(n), \hat{\mu}^*(n))$ has to be computed at each round. We can compute it by solving the dual problem discussed in Section 4 with e.g. Newton’s method. We omit detailed description of our implementation of DMED but we note that D_{\min} can be computed (or approximated) efficiently as follows:

- (1) The optimal solution $\nu^*(\hat{F}_i(n-1), \hat{\mu}_i(n-1))$ of the previous round is a good approximation of the current $\nu^*(\hat{F}_i(n), \hat{\mu}_i(n))$ and the iteration for the optimization halts quickly.
- (2) $\hat{\mu}^*(n)$ does not deviate significantly from μ^* for sufficiently large n and $D_{\min}(F, \mu)$ is differentiable in μ from Theorem 6. Therefore, $D_{\min}(\hat{F}_i, \hat{\mu}^*)$ can be approximated accurately by the linear approximation on $\hat{\mu}^*$ as long as \hat{F}_i is not updated. On the other hand, \hat{F}_i is updated only $O(\log n)$ times through n rounds and its effect on the complexity is small.

Each plot is an average over 1,000 different runs. The labels of each figure are as follows. “regret” denotes $\sum_{i:\mu_i < \mu^*} (\mu^* - \mu_i) T_i(n)$, which is the loss due to choosing suboptimal arms. “Dmin” stands for the asymptotic bound for a consistent policy, $\sum_{i:\mu_i < \mu^*} (\mu^* - \mu_i) \log n / D_{\min}(F_i, \mu^*)$. The asymptotic slope of the regret (in the semi-logarithmic plot) of a consistent policy is more than or equal to that of “Dmin”.

Figure 1 is a result for five arms with beta distributions. Beta distribution is an example of a simple continuous distribution on $[0, 1]$. Parameters for beta distributions are $(0.9, 0.1)$, $(7, 3)$, $(0.5, 0.5)$, $(3, 7)$, $(0.1, 0.9)$ and expectations are $\mu_i = 0.9, 0.7, 0.5, 0.3, 0.1$. Figure 2 is a result for two arms with discrete distributions

$$\begin{aligned} F_1(\{0\}) &= 0.99, & F_1(\{1\}) &= 0.01, & \mu_1 &= 0.01, \\ F_2(\{0.008\}) &= 0.5, & F_2(\{0.009\}) &= 0.5, & \mu_2 &= 0.0085. \end{aligned}$$

It is an example of a problem where the optimal arm is hard to distinguish since the suboptimal arm appears to be optimal at first with high probability. We see from these figures that DMED achieves a regret near the asymptotic bound.

7 Conclusion

We proposed a policy, DMED, and proved that our policy achieves the asymptotic bound for bounded support models. We also showed that our policy can be implemented efficiently by a convex optimization technique.

There are many models that D_{\min} can be computed explicitly, such as normal distribution model with unknown mean and variance. We expect that our DMED can be extended to these models.

It is also important to consider the finite horizon case and to derive a finite-time bound of DMED. A finite-time bound may be derived by a non-asymptotic form of Sanov’s Theorem in Exercise 6.2.19 of Dembo and Zeitouni (1998). However, its naive application makes the whole discussion extremely longer, e.g. the continuity of $D_{\min}(F, \mu)$ in F has to be of the form “if $L(F, F') \leq \epsilon$ then $|D_{\min}(F, \mu) - D_{\min}(F', \mu)| \leq \delta(\epsilon, F, \mu)$ ” with explicit $\delta(\cdot, \cdot, \cdot)$. Therefore other approaches may be more realistic.

References

- Agrawal, R. (1995). The continuum-armed bandit problem. *SIAM J. Control Optim.*, 33, 1926–1951.
- Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47, 235–256.
- Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2003). The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32, 48–77.
- Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.
- Burnetas, A. N., & Katehakis, M. N. (1996). Optimal adaptive policies for sequential allocation problems. *Adv. Appl. Math.*, 17, 122–142.
- Dembo, A., & Zeitouni, O. (1998). *Large deviations techniques and applications*, vol. 38 of *Applications of Mathematics*. New York: Springer-Verlag. Second edition.
- Even-Dar, E., Mannor, S., & Mansour, Y. (2002). Pac bounds for multi-armed bandit and markov decision processes. *Proceedings of COLT 2002* (pp. 255–270). London, UK: Springer-Verlag.
- Fiacco, A. V. (1983). *Introduction to sensitivity and stability analysis in nonlinear programming*. New York: Academic Press.
- Gittins, J. C. (1989). *Multi-armed bandit allocation indices*. Wiley-Interscience Series in Systems and Optimization. Chichester: John Wiley & Sons Ltd. With a foreword by Peter Whittle.
- Honda, J., & Takemura, A. (2010). An asymptotically optimal policy for finite support models in the multi-armed bandit problem. Submitted to *Machine Learning*, arXiv:0905.2776v3.
- Ishikida, T., & Varaiya, P. (1994). Multi-armed bandit problem revisited. *J. Optim. Theory Appl.*, 83, 113–154.
- Kleinberg, R. (2005). Nearly tight bounds for the continuum-armed bandit problem. *Proceedings of NIPS 2005* (pp. 697–704). MIT Press.
- Kleinberg, R. D., Niculescu-Mizil, A., & Sharma, Y. (2008). Regret bounds for sleeping experts and bandits. *Proceedings of COLT 2008* (pp. 425–436).
- Lai, T. L., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6, 4–22.
- Lamperti, J. (1996). *Probability; a survey of the mathematical theory*. Wiley Series in Probability Statistics. New York: John Wiley & Sons Ltd. Second edition.
- Meuleau, N., & Bourgin, P. (1999). Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, 35, 117–154.
- Pinsker, M. S. (1964). *Information and information stability of random variables and processes (transl.)*. San Francisco: Holden Day.
- Rockafellar, R. T. (1970). *Convex analysis (Princeton Mathematical Series)*. Princeton University Press.
- Strens, M. (2000). A bayesian framework for reinforcement learning. *Proceedings of ICML 2000* (pp. 943–950). Morgan Kaufmann, San Francisco, CA.
- Vermorel, J., & Mohri, M. (2005). Multi-armed bandit algorithms and empirical evaluation. *Proceedings of ECML 2005* (pp. 437–448). Porto, Portugal: Springer.
- Yakowitz, S., & Lowe, W. (1991). Nonparametric bandit methods. *Ann. Oper. Res.*, 28, 297–312.

Learning with Global Cost in Stochastic Environments

Eyal Even-dar
Google Research
evendar@google.com

Shie Mannor*
Technion
shie@ee.technion.ac.il

Yishay Mansour†
Tel Aviv Univ.
mansour@cs.tau.ac.il

Abstract

We consider an online learning setting where at each time step the decision maker has to choose how to distribute the future loss between k alternatives, and then observes the loss of each alternative, where the losses are assumed to come from a joint distribution. Motivated by load balancing and job scheduling, we consider a global cost function (over the losses incurred by each alternative), rather than a summation of the instantaneous losses as done traditionally in online learning. Specifically, we consider the global cost functions: (1) the makespan (the maximum over the alternatives) and (2) the L_d norm (over the alternatives) for $d > 1$. We design algorithms that guarantee logarithmic regret for this setting, where the regret is measured with respect to the best static decision (one selects the same distribution over alternatives at every time step). We also show that the least loaded machine, a natural algorithm for minimizing the makespan, has a regret of the order of \sqrt{T} . We complement our theoretical findings with supporting experimental results.

1 Introduction

Consider a decision maker that has to repeatedly select between multiple actions, while having uncertainty regarding the results of its action. This basic setting motivated a large body of research in machine learning, as well as theoretical computer science, operation research, game theory, control theory and elsewhere. Online learning in general, and regret minimization in particular, focus on this case. In regret minimization, in each time step the decision maker has to select between N actions, and only then observes the loss of each action. The cumulative loss of the decision maker is the sum of its losses at the various time steps. The main goal of regret minimization is to compare the decision maker's cumulative loss to the best strategy in a benchmark class, which many times is simply the set of all actions (i.e., each strategy will play the same action in every time step). The regret is the difference between the performance of the decision maker and the best strategy in the benchmark class. The main result is that if we allow the decision maker to play a mixture of the actions, the decision maker can guarantee to almost match the best single action even in the case that the losses are selected by an adversary: the regret would be of the order of $O(\sqrt{T \log N})$. (See [3] for an excellent exposition of the topic.)

In this work we are interested in extending the regret minimization framework to handle the case where the global cost is not simply additive across the time steps as was initiated in [4] for adversarial environments. The best motivating examples are load balancing and job scheduling. Assume that each action is a machine, and at each time step we need to map an incoming task to one of the machines, without knowing the cost that it will introduce on each machine (where the cost can be a function of the available resources on the machine and the resources the task requires). In such a setting we are interested in the load of each machine, which is the sum of the costs of the tasks map to it. A natural measure of the imbalance between the machines (actions) is either the makespan (the maximum load) or the L_d norm of the loads, both widely studied in

*This research was partially supported by the Israel Science Foundation under contract 890015 and by a Horev Fellowship and the EU under a Reintegration Grant.

†This work was supported in part by a grant from the Ministry of Science grant No. 3-6797, by a grant from the Israel Science Foundation (grant No. 709/09) and grant No. 2008-321 from the United States-Israel Binational Science Foundation (BSF), and by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication reflects the authors' views only.

job scheduling.¹ This setup, in the adversarial model where losses are assumed to be a deterministic (but unknown) sequence that is possibly even generated by an adversary, was introduced and studied in [4]. It was proved that a regret of the order of $O(\sqrt{TN})$ can be guaranteed where T is the time horizon and N is the number of actions or machines. For the specific case of makespan an improved regret of $O(\log N \sqrt{T})$ can be achieved.

In our model losses are generated from a joint distribution D . After every stage the decision maker observes the sampled loss vector and we are interested in both cases of full and partial information (that is: both cases where D is known and D is not known are of interest). The main result of our work is to show a logarithmic regret bounds for general cost functions (makespan and L_d norm). Contrary to most bandits setups, the distribution D can have arbitrary correlations between the losses of various actions, capturing the idea that some instances are inherently harder or easier. Note that the decision maker observes the entire loss vector, and thus our work is in the perfect observation model. We emphasize that the regret is never negative, and in the case that the decision maker global cost is less than the static optimum global cost then the regret is zero. Namely, the runs in which the decision maker outperforms the best static strategy are regraded as having zero regret.

To better understand our setting and results, it would be helpful to consider the following example where there are two actions and the global cost function is the makespan. The distribution D with probability half returns the loss vector $(0, 1)$, where action 1 has zero loss and action 2 has a loss of one, and with probability half D returns the loss vector $(1, 0)$. A realization of D of size T , will have $T/2 + \Delta$ losses of type $(0, 1)$ and $T/2 - \Delta$ losses of type $(1, 0)$. The most natural strategy the decision maker can use is to fix the best static strategy for D , and use it in all time steps. In this case the best strategy would be $(1/2, 1/2)$, and if there are $T/2 + \Delta$ losses of type $(0, 1)$ and $T/2 - \Delta$ losses of type $(1, 0)$ then the load on action 1 would be $T/4 + \Delta/2$, the load on action 2 would be $T/4 - \Delta/2$ and the makespan would be $T/4 + |\Delta|/2$. One can show that the best static strategy in hindsight would have both actions with the same load and both loads would be at most $T/4$. Since we expect that $|\Delta|$ be of the order of \sqrt{T} , this would give a regret bound of $\Theta(\sqrt{T})$. At the other extreme we can use a dynamic strategy that greedily selects at each time step the action with the lower load. This is the well-known *Least Loaded Machine* (LLM) strategy. For the analysis of the LLM we can consider the sum of the loads, since in the LLM strategy the max is just half of the sum for two machines. Since at each time step the LLM strategy selects deterministically an action, the sum of loads would be the sum of T Bernoulli random variables, which would be $T/2 + \Delta$. Since with constant probability we have $\Delta = \Theta(\sqrt{T})$, the regret would be $\Omega(\sqrt{T})$. The starting point of this research is whether the decision maker can do better than $\Theta(\sqrt{T})$ regret? The main result of this work is an affirmative answer to this question, showing logarithmic regret bounds.

In this work we consider two stochastic models, in the *known distribution model* the distribution D is known to the decision maker, while in the *unknown distribution model* the decision maker only knows that there is some distribution D that generates the examples. We consider two global cost function, the makespan, where the global cost is the maximum load on any action, and L_d norm for $d > 1$, where the global cost is an L_d norm of the loads. For both the makespan and the L_d norm we devise algorithms with a regret bound of $O(\log T \log \log T)$. In the unknown distribution model we show a regret bound of $O(\log^2 T \log \log T)$. The above regret bounds depend on knowing the exact number of time steps T in advance, and hold only at the last time step. We define *anytime regret* to be the case where we are given a bound T on the number of time steps and the regret bound has to hold at any time $t < T$. We present an algorithm with an anytime regret bound of $O(T^{1/3})$. We analyze the LLM strategy, showing that it has an anytime regret upper of $O(\sqrt{T} \log T)$ and lower bound of $\Omega(\sqrt{T})$. We also perform experiments that support our theoretical finding, and show the benefits of the algorithms that we developed.

It would be instructive to compare our stochastic model and results to other regret minimization models in stochastic environments under additive loss. First, note that in the known distribution model, the best action is known, and therefore the optimal algorithm for the additive loss would simply select the best action in every time step. For the makespan or L_d norm global cost functions, the online algorithm needs to compensate for the stochastic variations in the loss sequence, even in the known distribution model. Second, in the unknown distribution model, when the decision maker observes all the losses, i.e., the perfect observation model, the simple greedy algorithm is optimal. The greedy algorithm for makespan is the LLM, and we show that its regret is at least $\Omega(\sqrt{T})$ and at most $O(\sqrt{T} \log T)$. Finally, most of the work regarding stochastic environments was devoted to the multi-armed bandit problem, where the key issue is partial observation (which induces the *exploration vs exploitation tradeoff*): the decision maker observes *only* the loss of the action (arm) it chooses. The multi-armed bandit has been studied since [6] with the main result being logarithmic regret ([5] and [1]),

¹We remark that our information model differs from the classical job scheduling model in that we observe the costs only after we select an action, while in job scheduling you first observe the costs and only then select the action (machine).

with a constant that is a function of the distribution.

Regret for stochastic environments with memory has been studied in [2, 7] in the context of Markov decision problems (MDPs). The algorithms developed there obtain logarithmic regret but requires a finite state space and require that the MDP is either unichain or irreducible. Our problem can also be modelled as an MDP where the states are the load vector (or their differences) and the costs are additive. The state space in such a model is, however, continuous² and is neither unichain nor irreducible. Moreover, efficient exploration of the state space which is the hallmark of these works is not really relevant to online learning with global cost that is more focused on taking advantage of the local stochastic deviations from expected behavior.

2 Model

We consider an online learning setup where a scheduler has a finite set $N = \{1, \dots, n\}$ of n actions (machines) to choose from. At each time step $t \in [1, T]$, the scheduler A selects a distribution $\alpha_t^A \in \Delta(N)$ over the set of actions (machines) N , where $\Delta(N)$ is the set of distributions over N . Following that a vector of losses (loads) $\ell_t \in [0, 1]^n$ is drawn from a fixed distribution D , such that $p(i) = \mathbb{E}[\ell_t(i)]$ and $p_{\min} = \min_{i \in N} p(i)$. We consider both cases where D is known and unknown. We stress that D is an arbitrary distribution over $[0, 1]^n$, and can have arbitrary correlations between the losses of different actions. We do assume that loss vectors of different time steps are drawn independently from D .

Our goal is to minimize a given global cost function C which is defined over the average loss (or load) of each action. In order to define this more formally we will need to introduce a few notations. Denote the average loss (or load) of the online scheduler A on action (machine) i by $L_T^A(i) = \frac{1}{T} \sum_{t=1}^T \alpha_t^A(i) \ell_t(i)$ and its average loss (or load) vector is $L_T^A = (L_T^A(1), \dots, L_T^A(n))$.

We now introduce global cost functions. In this work we consider two global cost functions, the makespan, i.e., $C_\infty(x) = \max_{i \in N} x(i)$ or the L_d norm, i.e., $C_d(x) = (\sum_{i \in N} x(i)^d)^{1/d}$ for $d > 1$. This implies that the objective of the online scheduler A is to minimize either the *makespan*, i.e., $C_\infty(L_T^A) = \max_{i \in N} L_T^A(i)$ or the L_d norm, i.e., $C_d(L_T^A) = (\sum_{i \in N} (L_T^A(i))^d)^{1/d}$. Note that both the makespan and the L_d norm introduce a very different optimization problem in contrast to traditional online learning setup (adversarial or stochastic) where the cost is an additive function, i.e., $\sum_{i=1}^n L_T^A(i)$.

In order to define a regret we need to introduce a comparison class. Our comparison class is the class of static allocations for $\alpha \in \Delta(N)$. Again, we need to first introduce a few notations. Denote the average loss (or load) of machine i by $L_T(i) = \frac{1}{T} \sum_{t=1}^T \ell_t(i)$. The loss vector of a static allocation $\alpha \in \Delta(N)$ is $L_T^\alpha = \alpha \odot L_T$ where $x \odot y = (x(1)y(1), \dots, x(n)y(n))$. We define the *optimal cost function* $C^*(L_T)$ as the minimum over $\alpha \in \Delta(N)$ of $C(L_T^\alpha)$ and denote by $\alpha_C^*(L_T)$ a minimizing $\alpha \in \Delta(N)$, called the *optimal static allocation*, i.e.,

$$C^*(L_T) = \min_{\alpha \in \Delta(N)} C(L_T^\alpha) = \min_{\alpha \in \Delta(N)} C(\alpha \odot L_T).$$

For the makespan we denote the optimal cost by C_∞^* and by α_∞^* the optimal static allocation, i.e., we have

$$C_\infty^*(L_T) = \min_{\alpha \in \Delta(N)} \max_{i \in N} \alpha(i) L_T(i) = \max_{i \in N} \alpha_\infty^*(i) L_T(i).$$

Similarly for the L_d -norm we denote the optimal cost by C_d^* and by α_d^* .

As we mentioned before, We distinguish between two cases. In the *known distribution* model the scheduler A has as an input $p = (p(1), \dots, p(n))$, the expected loss of each action, while in the *unknown distribution* model A has no information (and has to estimate these quantities from data). We stress that in the known distribution model the online algorithm does not know the realization of the losses but has access only to the expectation under the distribution D .

The regret of scheduler A at time T after the loss sequence L_T is defined as,

$$R_T(L_T, A) = \max\{C(L_T^A) - C^*(L_T), 0\}.$$

The following claim proved in [4], prescribes the static optimum of makespan and L_d norm explicitly.

Claim 1 ([4]) *For the makespan global cost function, we have $\alpha_\infty^*(L_T) = (\frac{1/L_T(i)}{\sum_{i=1}^k 1/L_T(j)})_{i \in K}$ and $C_\infty^*(L_T) = (\frac{1}{\sum_{j=1}^k 1/L_T(j)})$. For the L_d norm we have $C_d^*(L_T) = (\frac{1}{\sum_{j=1}^k 1/L_j^{\frac{d-1}{d}}})^{\frac{d-1}{d}}$ and $\alpha_d^*(L_T) = (\frac{1/L_T(i)^{\frac{d-1}{d}}}{\sum_{j=1}^k 1/L_T(j)^{\frac{d-1}{d}}})_{i \in K}$.*

The functions C_∞ and C_d are convex and the functions C_∞^ and C_d^* are concave.*

²Even if one discretizes the MDP, the size will grow with time.

The following lemma is a standard concentration bound.

Lemma 2 (concentration bound - additive) *Let Z_1, \dots, Z_m be i.i.d. random variable in $[0, 1]$ with mean μ . Then, for any $\gamma > 0$, $\Pr \left[\left| \frac{1}{m} \sum_{i=1}^m Z_i - \mu \right| > \gamma \right] \leq 2e^{-2\gamma^2 m}$*

3 A Generic Load Balancing Algorithm

In this section we describe a generic load balancing algorithm G with low regret. The algorithm is described in a parametrized way. Later on we will provide concrete instances of G with regret rates for both of known and unknown distribution models and for the makespan and the L_d -norm global cost functions.

3.1 Overview

The basic idea of the generic algorithm is to partition the time in to m phases, where the length of phase k is T^k time steps.

In the known distribution model it is tempting to use always the optimal weights w^* derived from the expectations, essentially assuming that the realization will exactly match the expectation. In this case the regret would depend on the difference between the realization and the expectation. Such a strategy will yield a regret of the order of $O(\sqrt{T})$, which is the order of the deviations we are likely to observe between the realization and the expectation of the losses. Our main goal is to have a much lower regret, namely a logarithmic regret.

The main idea is the following. We start with the base weights derived from w^* . In phase k we perturb the base weights w^* depending on the deviation between w^* and opt^{k-1} , the optimal weights given the realization in phase $k-1$. At first sight it could appear counterintuitive that we can use the optimal allocation opt^{k-1} of phase $k-1$ to perturb the weights in phase k . Essentially, the optimal allocations in different phases are independent, given the known loss distribution D . However, consider the makespan for illustration, then any suboptimal allocation will have the load of some actions strictly larger than the loads of other actions. This suggests that for the actions with observed lower loads we can increase the weight, since we are concerned only with the action whose load is maximal. Essentially, our perturbations attempt to take advantage of those imbalances in order to improve the performance of the online algorithm and make it match better the optimal static allocation.

3.2 Generic Load Balancing Algorithm

The generic algorithm G depends on the following parameters: (1) C and C^* , the global cost and the optimal cost functions, respectively; (2) the number of phases m ; (3) the phases' lengths, (T^1, \dots, T^m) where T^k is the length of phase k ; and (4) $w^* \in \Delta(N)$ which is a distribution over the actions N .

The generic algorithm G runs in m phases where the length of the k -th phase is T^k . Let $q^k(i)$ be the observed average loss of action i at phase k , i.e., $q^k(i) = \sum_{t \in T^k} \ell_t(i) / T^k$. Let $opt^k(i)$ be the weight of action i in the optimal allocation for phase k and let $OPT^k(i) = opt^k(i)q^k(i)T^k$ be the load on action i in phase k using the optimal allocation for phase k .

The generic algorithm G has a parameter $w^* \in \Delta(N)$ that is the base weight vector (different applications of G will use different base weights w^*). During phase k the weight of action i the algorithm does not change, and it equals

$$w^k(i) = w^*(i) + \frac{1}{q^{k-1}(i)} \frac{OPT^{k-1}(i) - X^{k-1}(i)}{T^k} = w^*(i) + \frac{T^{k-1}}{T^k} (opt^{k-1}(i) - w^*(i)), \quad (1)$$

where $X^{k-1}(i) = w^*(i)q^{k-1}(i)T^{k-1}$ if all $w^k(i)$'s are positive. Otherwise we set $w^k(i) = w^*$. First note that G has all the information to compute the weights. Second, note that $X^{k-1}(i)$ depends on $w^*(i)$ and not on $w^{k-1}(i)$. Third, at first sight it might look that $w^k(i)$ and $w^*(i)$ can be very far apart, however, in the analysis we will require that $opt^{k-1}(i)$ and $w^*(i)$ are close, and therefore $w^k(i)$ and $w^*(i)$ will be close.

3.3 Analysis of the Generic Algorithm

We now turn to deriving the properties of the generic algorithm G that will be used later for specific setups. Before we start the analysis of the generic algorithm G we would like to state a few properties that will be essential to our analysis. Some of the properties depend on the realized losses while other depend on the behaviour of the algorithm G and its parameters.

Definition 3 *Phase k of an algorithm is (α, β) -opt-stable if it has the following properties:*

P1 For any action i we have that $|q^k(i) - p(i)| \leq \beta / \sqrt{T^k}$ and that $q^k(i) > 0$;

P2 For any action i we have that $|w^(i) - opt^k(i)| = \epsilon^k(i)$, where $\epsilon^k(i) < \alpha / \sqrt{T^k}$; and*

P3 The phase lengths satisfies that $T^{k-1} \leq 4T^k$.

Let us explain and motivate the above properties. Property P1 is a property of the realization of the losses in a phase, and it states that the empirical frequency of the losses are close their true expectations. Property P3 depends solely on the parameters of G and relates the length of adjacent phases, requiring that the length does not shrink too fast. Property P2 requires the base weights to be close to the optimal weight for all actions. In this section we assume that all properties hold, while for each instance of G we will need to show that they indeed hold for most of the phases, with high probability, for the specified parameters.

The first step in the analysis is to show that the weights assigned by the generic algorithm G are indeed valid. First note that if we had a negative value in Eq. (1) (i.e., $w^*(i) + \frac{T^{k-1}}{T^k}(opt^{k-1}(i) - w^*(i)) < 0$), then we set $w^k(i) = w^*$ and the weights are valid by definition. Claim 4 shows that the weights always sum to 1. Claim 5 shows that Eq. (1) is non-negative if the phase is (α, β) -opt-stable.

Claim 4 For any phase k we have $\sum_{i \in N} w^k(i) = 1$.

Proof: First note that if we set $w^k = w^*$ then the the claim holds. The proof follows from the following identities.

$$\begin{aligned} \sum_{i \in N} w^k(i) &= \sum_{i \in N} \left[w^*(i) + \frac{T^{k-1}}{T^k}(w^*(i) + \sum_{i \in N} opt^{k-1}(i) - w^*(i)) \right] \\ &= 1 + \frac{T^{k-1}}{T^k} \sum_{i \in N} opt^{k-1}(i) - \frac{T^{k-1}}{T^k} \sum_{i \in N} w^*(i) = 1. \quad \blacksquare \end{aligned}$$

The following claim shows that $w^k(i)$ in Eq. (1) are non-negative and close to the base weights $w^*(i)$.

Claim 5 If phase k is (α, β) -opt-stable and $T_k > (\frac{4\alpha}{w^*(i)})^2$ then for every action i , we have

$$|w^k(i) - w^*(i)| \leq \frac{T^{k-1}}{T^k} \frac{\alpha}{\sqrt{T^k}} \leq \frac{4\alpha}{\sqrt{T^k}}.$$

Proof: First note that if we set $w^k(i) = w^*(i)$ then the the claim holds. Otherwise

$$w^k(i) \geq w^*(i) - \frac{T^{k-1}}{T^k} |opt_i^{k-1} - w^*(i)| \geq w^*(i) - \frac{T^{k-1}}{T^k} \frac{\alpha}{\sqrt{T^k}} \geq 0,$$

where the first inequality uses P2 and the last uses property P3 that $T^{k-1}/T^k \leq 4$ and the fact that $T_k > (4\alpha/w^*(i))^2$. \blacksquare

The most important observation is made in the next lemma that considers the increase in the load due to the generic algorithm G . It shows that the increase in the load of action i in phase k can be decomposed to three parts. The first is the optimal cost of action i in the previous phase, phase $k - 1$. The second is the difference between the load of the base weight in phase k and $k - 1$. The third can be viewed as a constant, and will contribute at the end to the regret.

Lemma 6 Suppose that phase k is (α, β) -opt-stable. Then the increase for action i in the phase is bounded by

$$OPT^{k-1}(i) + X^k(i) - X^{k-1}(i) + \alpha\beta \left(1 + \sqrt{\frac{T^{k-1}}{T^k}} \right).$$

Proof: The increase of the load of the online algorithm for action i during phase k is $w^k(i)q^k(i)T^k$. Now,

$$\begin{aligned} w^k(i)q^k(i)T^k &= w^*(i)q^k(i)T^k + \frac{q^k(i)}{q^{k-1}(i)}(OPT^{k-1}(i) - X^{k-1}(i)) \\ &= X^k(i) + \left(1 + \frac{q^k(i) - q^{k-1}(i)}{q^{k-1}(i)}\right)(OPT^{k-1}(i) - X^{k-1}(i)) \\ &= X^k(i) + OPT^{k-1}(i) - X^{k-1}(i) + \frac{q^k(i) - q^{k-1}(i)}{q^{k-1}(i)}(OPT^{k-1}(i) - X^{k-1}(i)) \\ &= X^k(i) + OPT^{k-1}(i) - X^{k-1}(i) + (q^k(i) - q^{k-1}(i))(opt_i^{k-1} - w^*(i))T^{k-1} \\ &= X^k(i) + OPT^{k-1}(i) - X^{k-1}(i) + (q^k(i) - q^{k-1}(i))\epsilon^{k-1}(i)T^{k-1} \\ &\leq X^k(i) + OPT^{k-1}(i) - X^{k-1}(i) + (\beta/\sqrt{T^k} + \beta/\sqrt{T^{k-1}})(\alpha/\sqrt{T^{k-1}})T^{k-1} \\ &= X^k(i) + OPT^{k-1}(i) - X^{k-1}(i) + \alpha\beta + \alpha\beta\sqrt{\frac{T^{k-1}}{T^k}}, \end{aligned}$$

where the inequality is due to properties $P1$ and $P2$. ■

The following theorem summarizes the performance of the generic algorithm G , showing that its regret depends on two parts, the regret of the base weights in the last phase and a term which is linear in the number of phases.

Theorem 7 *Assume that C^* is concave, C is convex, and $C(a, \dots, a) = a$ for $a > 0$. Suppose that phases $1 \dots m'$ of the generic algorithm G with its parameters are (α, β) -**opt-stable** and that $T^{m'} \geq \max_i (4\alpha/w^*(i))^2$. Then its cost in these phases is bounded by*

$$OPT + R^{m'} + 3m'\alpha\beta,$$

where OPT is the optimal cost, $R^{m'} = \max_i R^{m'}(i)$ and $R^{m'}(i) = \max\{X^{m'}(i) - OPT^{m'}(i), 0\}$. Furthermore, if $m' < m$ then its cost is bounded by

$$OPT + 3m'\alpha\beta + \sum_{k=m'}^m R^k.$$

Proof: Since C is convex, for any $Z > 0$

$$C\left(\sum_{k=1}^m OPT^k(1) + Z, \dots, \sum_{k=1}^m OPT^k(N) + Z\right) \leq \sum_{k=1}^m C(OPT^k(1), \dots, OPT^k(N)) + Z.$$

Let L^k be the losses in phase k . By definition, $C(OPT^k(1), \dots, OPT^k(N)) = C^*(L^k)$. Since C^* is concave

$$\sum_{k=1}^m C^*(L^k) \leq C^*\left(\sum_{k=1}^m L^k\right) = OPT.$$

We first deal with regret in the first m' rounds. By Lemma 6, the increase of load at the k th phase is bounded by

$$X^k(i) + OPT^{k-1}(i) - X^{k-1}(i) + \alpha\beta\left(1 + \sqrt{\frac{T^{k-1}}{T^k}}\right) \leq X^k(i) + OPT^{k-1}(i) - X^{k-1}(i) + 3\alpha\beta.$$

Summing over the different phases we obtain that the loss of G on action i is at most

$$\left[\sum_{k=1}^{m'} OPT^k(i) + 3\alpha\beta\right] + X^{m'}(i) - OPT^{m'}(i).$$

Applying the cost C to this we bound the cost on the G online algorithm by

$$C(L_T^G) \leq OPT + R^{m'} + 3m'\alpha\beta.$$

For the last $m - m'$ phases since C is convex and C^* is concave the total regret is bounded by the sum of the regrets over the phases completing the proof of the second part of the theorem. ■

We can summarize the theorem in the following corollary, for the cases that are of interest to us, the makespan and the L_d norm.

Corollary 8 *Assuming that the first m' phases of the generic algorithm G with its parameters are (α, β) -**opt-stable** with probability at least $1 - \delta$ and $T^{m'} \geq \max_i (4\alpha/w^*(i))^2$ for the makespan and L_d norm, then its regret is bounded by*

$$3m\alpha\beta + \sum_{k=m'}^m T^k,$$

with probability at least $1 - \delta$.

4 Makespan

In this section we consider the makespan as the global cost function. We first analyze the case of where the distribution is known and then turn to the case of where the distribution is unknown.

4.1 Known distribution

To apply the generic algorithm G we need to specify its parameters: (1) the number of phases and their duration, and (2) the base weights.

- Set $w^*(i) = \frac{1/p(i)}{P}$, where $P = \sum_{i=1}^n 1/p(i)$, i.e., the optimal allocation for p .
- Set the number of phases $m = \log(T)$.
- Set the length of phase k to be $T^k = T/2^k$ for $k \in [1, m]$.

Let G_∞^{kn} be the generic algorithm G with the above parameters and the cost functions makespan.

The analysis divides the phases into two sets: we show that the first $\log T/A$ phases, where A will be defined shortly, are (α, β) -**opt-stable** with high probability; for the remaining phases we show that $\sum_{k=\log T/A}^m T^k$ is small; the result would follow from Corollary 8.

Let $A = \lceil \max\{4\beta^2/p_{\min}^2, \max_i 4\alpha^2/w^*(i)^2\} \rceil$, $\beta = \sqrt{(1/2)\ln(1/\eta)}$, and $\alpha = 6\beta/(Np_{\min}^4)$, where η is a parameter that controls the success probability. Since for the first $\log T/A$ phases we have $T^k > 4\alpha^2/w^*(i)^2$, once we show that with high probability all the first $\log T/A$ phases are (α, β) -**opt-stable**, we establish the following theorem.

Theorem 9 *Suppose we run Algorithm G_∞^{kn} with parameters $\alpha = 6\beta/(Np_{\min}^4)$, $\beta = \sqrt{(1/2)\ln(1/\eta)}$, and $\eta = \delta/Nm$. Then with probability at least $1 - \delta$ the regret is bounded by*

$$O\left(\log(T) \log\left(\frac{N \log T}{\delta}\right) \frac{1}{Np_{\min}^4} + \frac{1}{p_{\min}^{10}} \log\left(\frac{N \log T}{\delta}\right)\right) = O(\log T \log \log T).$$

The next sequence of lemmas show that the first phases of G_∞^{kn} is (α, β) -**opt-stable**, where the last lemma bounds the contribution from the last phases.

The simplest claim is showing that P3 holds, which is immediate from the definition of T^k .

Claim 10 G_∞^{kn} satisfies property P3.

The additive concentration bound (Lemma 2) together with the fact that $T^k \geq \max\{4\beta^2/p_{\min}^2, 4\alpha^2/w^*(i)^2\}$ for the first $\log T/A$ phases imply that for these phases property P1 is satisfied with high probability.

Claim 11 *With probability $1 - mN\eta$, for any phase $k < \log T/A$ and action i we have, $|q^k(i) - p(i)| < \beta/\sqrt{T^k}$ and $q^k(i) > 0$ where $\beta = \sqrt{(1/2)\ln(1/\eta)}$.*

The following lemma proves that property P2 is satisfied.

Lemma 12 *With probability $1 - Nm\eta$, for any phase $k < \log T/A$ and action i ,*

$$|w^*(i) - opt^k(i)| \leq \frac{\alpha}{\sqrt{T^k}},$$

where $\beta = \sqrt{\frac{1}{2}\ln(1/\eta)}$ and $\alpha = 6\beta/(Np_{\min}^4)$.

Proof: First,

$$\left| \frac{1}{p(i)} - \frac{1}{q^k(i)} \right| = \frac{|p(i) - q^k(i)|}{p(i)q^k(i)} \leq \frac{\beta}{\sqrt{T^k}} \frac{2}{(p(i))^2},$$

where we used the bound on $|p(i) - q^k(i)|$ from Claim 11 and the fact that for the first $\log T/A$ phases we have $T^k \geq \max\{4\beta^2/p_{\min}^2, 4\alpha^2/w^*(i)^2\}$ implies that $q^k(i) \geq p(i) - \beta/\sqrt{T^k} \geq p(i)/2$ for these phases.

Second, we show that,

$$|P - Q^k| = \left| \sum_{i=1}^n \frac{1}{p(i)} - \frac{1}{q^k(i)} \right| \leq \sum_{i=1}^n \left| \frac{1}{p(i)} - \frac{1}{q^k(i)} \right| \leq N \frac{\beta}{\sqrt{T^k}} \frac{2}{p_{\min}^2}.$$

Since $1 \leq 1/p(i) \leq 1/p_{\min}$, we have that $N \leq P \leq N/p_{\min}$. By Claim 11 we have $|p(i) - q^k(i)| \leq \beta/\sqrt{T^k}$, and this implies that,

$$|p(i)P - q^k(i)Q^k| \leq P |p(i) - q^k(i)| + q^k(i) |P - Q^k| \leq \frac{N}{p_{\min}} \frac{\beta}{\sqrt{T^k}} + N \frac{\beta}{\sqrt{T^k}} \frac{2}{p_{\min}^2} \leq \frac{3N\beta}{p_{\min}^2 \sqrt{T^k}}.$$

Since $P \geq N$ and similarly $Q^k \geq N$, this implies that,

$$|w_i^* - \text{opt}^k(i)| = \left| \frac{1/p(i)}{P} - \frac{1/q^k(i)}{Q^k} \right| = \frac{|p(i)P - q^k(i)Q^k|}{p(i)Pq^k(i)Q^k} \leq \frac{2(|p(i)P - q^k(i)Q^k|)}{p_{\min}^2 N^2} = \frac{6\beta}{Np_{\min}^4 \sqrt{T^k}},$$

which establishes the lemma. \blacksquare

To establish the number of time steps in phases which are shorter than A , we need to lower bound $w^*(i)$.

Lemma 13 *For any action i we have $w^*(i) \geq p_{\min}/N$, which implies that the total time at the last $\log A$ phases is bounded by $O(A) = O(\log(1/\eta)/p_{\min}^{10})$.*

Proof: Since the length of the phases decays by factor of two, it is enough to bound $A = \max\{4\beta^2/p_{\min}^2, 4\alpha^2/w^*(i)^2\}$. Consider action i . We have that $P = \sum_{j=1}^n 1/p(j) \geq 1/p(i) + (1/p_{\min})(N-1)$. Thus,

$$w^*(i) = \frac{1/p(i)}{P} \geq \frac{1/p(i)}{1/p(i) + (N-1)/p_{\min}} = \frac{p_{\min}}{p(i)p_{\min} + p(i)(N-1)} \geq \frac{p_{\min}}{N}.$$

Substituting the value of α we obtain the lemma. \blacksquare

Proof of Theorem 9 : By Lemma 12 and Claims 11, 10 show the first $\log T/A$ phases are (α, β) -**opt-stable**. Lemma 13 bounds the length of the other phases. Applying Corollary 8 proves the theorem. \blacksquare

4.2 Unknown Distribution

The algorithm for an unknown distribution relies heavily on the application developed in the previous subsection for the known distribution model. We partition the time T to $\log(T/2)$ blocks, where the r -th block, B^r , has 2^r time steps. In block r we run G_{∞}^r , the generic algorithm G with the following parameters:

- Set $w^{r,*}(i)$ using the observed probabilities in block B^{r-1} as follows. Let $\text{opt}^{r-1}(i)$ be the optimal weight for action i in block B^{r-1} , then $w^{r,*}(i) = \text{opt}^{r-1}(i)$. (For $r = 1$ set $w^{1,*}$ arbitrarily, note that there are only two time steps in B^1 .)
- In block B^r we have $m = r$ phases, where the duration of phase k is $T^{r,k} = |B^r|/2^k = 2^{r-k}$.

We now need to show that the algorithm G_{∞}^r is (α, β) -**opt-stable**. We start by showing that property $P3$ is satisfied.

Claim 14 *With probability $1 - mN\eta$ we have that $|w^{r,*}(i) - \text{opt}^{r,k}(i)| \leq 2\alpha/\sqrt{T^{r,k}}$, where $\beta = \sqrt{\frac{1}{2} \ln(1/\eta)}$ and $\alpha = 6\beta/(Np_{\min}^4)$.*

Proof: Let $w^*(i) = (1/p(i))/P$. Then,

$$\begin{aligned} |w^{r,*}(i) - \text{opt}^{r,k}(i)| &= |\text{opt}^{r-1}(i) - \text{opt}^{r,k}(i)| \leq |\text{opt}^{r-1}(i) - w^*(i)| + |w^*(i) - \text{opt}^{r,k}(i)| \\ &\leq \frac{\alpha}{2^{(r-1)/2}} + \frac{\alpha}{2^{(r-k)/2}} \leq \frac{2\alpha}{\sqrt{T^{r,k}}}, \end{aligned}$$

where we used Lemma 12 twice for the second inequality. \blacksquare

Remember that $A = \max\{4\beta^2/p_{\min}^2, 4\alpha^2/w^*(i)^2\}$. Thus for all blocks with $B^r < A$, our regret bounds will be trivial. A more subtle point is that A is actually A_r as $w^*(i)$ changes between the different blocks. So we still need to compute A_r for any block to be able to compute the regret bounds. Note that by definition $P1$ and $P3$ are satisfied by the same reasoning as in the previous section,

Lemma 15 *If $B^{r-1} > 4\beta^2/p_{\min}^2$, then $w^{r,*} \geq p_{\min}/(2N)$ and $A_r = O(\log 1/\eta/p_{\min}^{10})$ with probability at least $1 - \eta$.*

Proof: Let \hat{q} be the average loss at the $r-1$ block. Since $B^{r-1} > 4\beta^2/p_{\min}^2$, then $\hat{q}_r(i) \geq p(i) - \beta/\sqrt{B^{r-1}} \geq p(i)/2$. Thus $1/\hat{q}_r(i) \leq 2/p_{\min}$. Thus

$$\frac{1/\hat{q}_r(i)}{\sum_{j \in N} 1/\hat{q}_r(j)} \geq \frac{1}{\sum_{j \in N} 1/\hat{q}_r(j)} \geq \frac{p_{\min}}{2N}.$$

The second of the part of the proof is identical to the proof in Lemma 13 \blacksquare

In each block B^r , Theorem 9 bounds the regret of G_{∞}^r , and we derive the following,

Lemma 16 *With probability $1 - rN\eta$, the regret during B^r is at most*

$$O\left(\log(|B^r|) \frac{\log(1/\eta)}{Np_{\min}^4} + \frac{1}{p_{\min}^{10}} \log(1/\eta)\right).$$

Summing over the m blocks we obtain the following theorem.

Theorem 17 *With probability $1 - \delta$, the regret is at most*

$$O\left(\log^2 T \log\left(\frac{N \log T}{\delta}\right) \frac{1}{p_{\min}^4} + \frac{\log T}{p_{\min}^{10}} \log\left(\frac{N \log T}{\delta}\right)\right) = O(\log^2 T \log \log T).$$

5 L_d norm

As in the makespan case (in the previous section) we apply the generic algorithm to the L_d norm global cost function. Note that the generic algorithm behaves differently for different global cost functions, even if the same parameters are used since OPT is different. From the implementation perspective, the only difference is that we set the base distribution w^* to the optimal one with respect to the L_d norm rather than the makespan. From the proof perspective, the key difference between the makespan and the L_d norm is the proof of property P2. Also, since the optimal allocation is different, we will have a different value of α . (The proofs of this section are omitted.)

5.1 Known distribution

In this section we define G_d^{kn} , which will have a low regret for the L_d norm in the known distribution model. Let G_d^{kn} be the generic algorithm G the following parameters.

- Set $w^*(i) = (p(i))^{-\lambda}/P_\lambda$, where $\lambda = d/(d-1)$ and $P_\lambda = \sum_{i=1}^n (p(i))^{-\lambda}$, i.e., the optimal allocation for p under the L_d norm.
- Set the number of phases $m = \log(T)$ and the length of phase k to be $T^k = T/2^k$ for $k \in [1, m]$.

The following lemma will be used to show property P2, and is similar to Lemma 12 for the makespan.

Lemma 18 *With probability $1 - Nm\eta$, for any action i and phase $k < \log(T/A)$ and $A = (4\beta^2/p_{\min}^2)$, we have*

$$|w^*(i) - opt^k(i)| \leq \frac{\alpha}{\sqrt{T^k}},$$

where $\beta = \sqrt{0.5 \ln(1/\eta)}$, and $\alpha = O\left(d\sqrt{\ln(1/\eta)}/\left((d-1)Np_{\min}^{4d/(d-1)}\right)\right)$.

To apply Corollary 8 we need to bound the sum of the lengths of phases of size less than A , which is done in the following lemma.

Lemma 19 *For any action i we have $w^*(i) \geq p_{\min}/N$, which implies that the total time in the last $\log A$ phases is bounded by $O(A) = O\left(d^2 \log(1/\eta)/\left((d-1)^2 p_{\min}^{10d/(d-1)}\right)\right)$.*

Similarly to the previous section we obtain the following.

Theorem 20 *Algorithm G_d^{kn} , with probability at least $1 - \delta$, has regret at most*

$$O\left(\log(T) \log\left(\frac{N \log T}{\delta}\right) \frac{d}{(d-1)Np_{\min}^{4d/(d-1)}} + \frac{d^2}{(d-1)^2 p_{\min}^{10d/(d-1)}} \log\left(\frac{N \log T}{\delta}\right)\right) = O(\log T \log \log T).$$

5.2 Unknown distribution

Similar to the makespan case, algorithm G_d^{un} , for L_d norm in the unknown distribution model, runs in blocks of increasing size, where in each block uses as the base distribution the optimal allocation for the previous block. The proofs are similar to the makespan case, where the differences are due to the different global cost function.

For algorithm G_d^{un} , we partition the time first to $\log(T/2)$ blocks, where the r -th block, B^r , has 2^r time steps. In block r we run G_d^r , the generic algorithm G with the following parameters:

- Set $w^{r,*}(i)$ using the observed probabilities in block B^{r-1} as follows. Let $opt^{r-1}(i)$ be the optimal weight for action i in block B^{r-1} , then $w^{r,*}(i) = opt^{r-1}(i)$. (For $r = 1$ set $w^{1,*}$ arbitrarily, note that there are only two time steps in B^1 .)

- In block B^r we have $m = r$ phases, where the duration of phase k is $T^{r,k} = |B^r|/2^k = 2^{r-k}$.

Similarly to the previous section we obtain the following.

Theorem 21 *Algorithm G_d^{un} , with probability $1 - \delta$, has regret at most*

$$O\left(\log^2 T \log\left(\frac{N \log T}{\delta}\right) \frac{d}{(d-1)p_{\min}^{4d/(d-1)}} + \frac{d^2 \log T}{(d-1)^2 p_{\min}^{10d/(d-1)}} \log\left(\frac{N \log T}{\delta}\right)\right) = O(\log^2 T \log \log T).$$

6 Any Time Regret

The regret algorithms presented so are optimizing the regret at the termination time T . It is not hard to see that the previous algorithms have regret of $O(\sqrt{T})$ at the initial part of the sequence. In this section we develop algorithms that have low regret at any time $t \in [1, T]$, and develop a different application of the generic algorithm which guarantees at an anytime a regret of $O(T^{1/3})$. We start from the case of known distribution and move to the case of an unknown one. (The proofs are omitted.)

6.1 Known Distribution

For the known distribution model we will describe two algorithms: G_∞^{any} for the makespan and G_d^{any} for the L_d norm. We set the parameters of the algorithms as follows:

- For G_∞^{any} set $w^*(i) = (p(i))^{-1}/P$, where $P = \sum_{i=1}^n (p(i))^{-1}$. For G_d^{any} set $w^*(i) = (p(i))^{-\lambda}/P_\lambda$, where $P_\lambda = \sum_{i=1}^n (p(i))^{-\lambda}$ and $\lambda = d/(d-1)$.
- For both G_∞^{any} and G_d^{any} set $m = T^{1/3}$, the number of phases, and let the phase length be constant: $T^k = T^{2/3}$ for $k \in [1, m]$.

The main result of this section is the following theorem.

Theorem 22 *Given the expected loss $p(i)$ of each action i , then with probability at least $1 - Nm\eta$, the regret of G_∞^{any} and G_d^{any} at any $t \in [1, T]$ is $O(\alpha_\infty T^{1/3})$ and $O(\alpha_d T^{1/3})$, respectively, where $\alpha_\infty = O(\sqrt{\log(1/\eta)}/(Np_{\min}^4))$, $\alpha_d = O(d\sqrt{\ln(1/\eta)}/((d-1)Np_{\min}^{4d/(d-1)}))$, and $\beta = O(\sqrt{\log(1/\eta)})$.*

When all the phases have identical length, using Eq. (1), we observe that $w^k(i)$ is simply $opt^{k-1}(i)$.

Observation 23 *If $T^k = T^{k-1}$ then $w^k(i) = opt^{k-1}(i)$.*

One advantage of the above observation is that we are guarantee that w^k is a distribution. Therefore, we can drop the requirement that the phase length is at least $4\alpha^2/w^*(i)^2$, which comes from Claim 5.

Claim 24 *Suppose that all phases have identical length $T^k > 4\beta^2/p_{\min}^2$. Then for $\beta = O(\sqrt{\log(1/\eta)})$, for the makespan $\alpha_\infty = O(\sqrt{\log(1/\eta)}/(Np_{\min}^4))$, for the L_d norm $\alpha_d = O(d\sqrt{\ln(1/\eta)}/((d-1)Np_{\min}^{4d/(d-1)}))$, then with probability at least $1 - \eta Nm$ all phases are (α, β) -opt-stable.*

The next lemma bounds the regret from the start of phase k until some time t in phase k , compared to the optimal allocation for this time period.

Lemma 25 *Suppose that the global cost C is convex and $C((a, \dots, a)) = a$ and that an algorithm is (α, β) -opt-stable algorithm at stages $1, 2, \dots, k$. Then, at any time t during phase k , the regret is at most $O(\alpha T^{1/3})$.*

Proof: By Observation 23 during phase k we use weight $w^k(i) = opt^{k-1}(i)$ for action i . Since all phases are (α, β) -opt-stable then this weight is close to the base weight, namely, we have that $|w^k(i) - w^*(i)| = |opt^{k-1}(i) - w^*(i)| \leq \alpha/\sqrt{T^k}$. Let $\tau = 1 + \sum_{j=1}^{k-1} T^j$ be the start of phase k . Let $opt^{k,t}(i)$ be the weight of the optimal allocation for the period starting at phase k and until time t , i.e., during time steps $[\tau, t]$. Similarly to the proofs of Claims 12 and 18, just for $t - \tau$ time steps (rather than T_k) we get that $|w^*(i) - opt^{k,t}(i)| \leq \alpha/\sqrt{t - \tau}$. Combining the two and using the fact that $T^k = T^{k-1}$ we obtain that $|w^k(i) - opt^{k,t}(i)| \leq 2\alpha/\sqrt{t - \tau}$. The difference between weight of the algorithm and the optimal allocation during $[\tau, t]$ in phase k , for any action i is at most $2\alpha(t - \tau)/\sqrt{t - \tau} \leq 2\alpha\sqrt{T^{2/3}}$. Since the global cost C is convex and $C((a, \dots, a)) = a$ the regret is at most $O(\alpha T^{1/3})$. ■

We are now ready to bound the anytime regret.

Proof of Theorem 22 : Consider a time t in phase k . By Claim 24, the algorithm is (α, β) -**opt-stable**. Applying Lemma 25 to phase $k - 1$ we have that we have that $R^{k-1} = O(\alpha_\infty T^{1/3})$ for makespan and $R^{k-1} = O(\alpha_d T^{1/3})$ for L_d norm. By Lemma 25 the regret in the period $[\tau, t]$ is at most $O(\alpha_\infty T^{1/3})$ for makespan and $O(\alpha_d t^{1/3})$ for L_d norm. We apply Corollary 8 to derive the theorem.³ ■

6.2 Unknown distribution

Similar to the case before, of the makespan and L_d norm, we use blocks of increasing size to (implicitly) estimate the expectation of the losses. Formally, we have $m = \log(T/2)$ blocks, where the r -th block, B^r , has 2^r time steps. In each block we run our G_∞^{any} and G_d^{any} algorithms. The parameters are:

- Set $w^{r,*}(i)$ to be the optimal allocation in the previous block, i.e., $opt^{k-1}(i)$, where in G_∞^{any} we use the makespan and in G_d^{any} we use the L_d norm.
- In block B^r we have $m = |B^r|^{1/3}$ phases, where the duration of phase k is $T^{r,k} = |B^r|^{2/3}$.

Let $A = 4\beta^2/p_{\min}^2$. For all blocks which are smaller than $A^{3/2}$ we simply bound their regret by their time. For the longer blocks, we show that in *each* block all phases are (α, β) -**opt-stable** (with high probability). Therefore, we obtain the following theorem.

Theorem 26 *With probability $1 - N\eta T^{1/3}$, at anytime the regret is at most $O(T^{1/3}\alpha + \log^{3/2} 1/\eta/p_{\min}^3) = O(T^{1/3})$, where $\alpha = O((\sqrt{\log(1/\eta)})/(Np_{\min}^4))$ for makespan and $\alpha = O(\frac{d\sqrt{\ln \frac{1}{\eta}}}{(d-1)Np_{\min}^{4d/(d-1)}})$ for L_d norm.*

7 Least Loaded Machine Scheduler

The Least load machine (LLM) scheduler is an intuitive and frequently used algorithm to minimize the makespan. The LLM scheduler puts all the weight of the next job on the least loaded machine machine, or, in our terminology, LLM selects the action with the least observed losses. The LLM scheduler is geared towards minimizing the makespan, and we will show that the regret of the LLM scheduler is at most $O(\sqrt{T} \log T)$ in the anytime model. On the other hand, the LLM scheduler suffers a regret which is $\Omega(\sqrt{T})$, a property that is common to all deterministic schedulers that allocate all the weight to a single machine.

In our setting all losses are bounded by 1 so at any time t the LLM scheduler will satisfy that the difference between the load of different actions is bounded by 1.

Lemma 27 *At time $t \in [t, T]$ the difference between the load of any two actions is bounded by 1, i.e., $|L_t^{LLM}(i) - L_t^{LLM}(j)| \leq 1$ for any $t \in [1, T]$ and $i \in N$.*

We prove in the following sequence of lemmas that the frequency of LLM using machine i is proportional to $1/p(i)$. The first step is to define when the realized losses are “representative” of the expectations. A realization of the losses is a matrix M of size $n \times T$, where the entry (i, k) is distributed according to $\ell(i)$ (using D). We can view the generation process of the losses as follows. The k -th time LLM uses action i we return the loss in entry $M[i, k]$. One can see that this gives an identical distribution to D .

Definition 28 *A realization matrix M is representative if for any i and k we have $\left| \sum_{j=1}^k M[i, j]/k - p(i) \right| \leq \sqrt{(\log 1/\eta)/(2k)}$.*

Using a simple concentration bound (Lemma 2) we have the following lemma.

Lemma 29 *With probability $1 - 2NT\eta$ the realization matrix M is representative.*

We are interested in cases where the loads on the actions is almost balanced. This definition will formally specify when an action selection vector results in an almost balanced loads.

Definition 30 *An integer vector $(k(1), \dots, k(N))$ is ℓ balanced for a matrix M if $\sum_{i=1}^N k(i) = T$ and for every i we have $\sum_{j=1}^{k(i)} M[i, j] \in [\ell, \ell + 1]$.*

The following lemma shows that if the loads are almost balanced then the makespan is close to T/P .

³Technically, the bound of the minimum phase required in Corollary 8 does not hold, but using Observation 23 we can derive an identical statement with an almost identical proof.

Lemma 31 Given a representative matrix M and an ℓ balanced vector $(k(1), \dots, k(N))$, we have that $|\ell - T/P| = O(\sqrt{T \log(1/\eta)})$.

Proof: Since M is representative, for each i we have that $|\sum_{j=1}^{k(i)} M[i, j] - p(i)k(i)| \leq \sqrt{0.5k(i) \log(1/\eta)}$. Since $(k(1), \dots, k(N))$ is ℓ balanced we have that $|\ell - p(i)k(i)| \leq \sqrt{0.5k(i) \log(1/\eta)} + 1$. Let $k(i) = \ell/p(i) + \lambda(i)/p(i)$, where we have shown that $|\lambda(i)| \leq \sqrt{0.5k(i) \log(1/\eta)} + 1$. Summing over all actions, we obtain that $T = \sum_{i=1}^N k(i) = \ell P + \Lambda$, where $\Lambda = \sum_{i=1}^N \lambda(i)/p(i)$. Therefore $T/P - \ell = \Lambda/P = \sum_{i=1}^N \lambda(i)(1/p(i))/P$. The lemma follows since $\Lambda/P < \max_i \lambda(i) \leq \sqrt{0.5T \log(1/\eta)} + 1$. ■

The next claim states that the LLM scheduler produces balanced vectors, given its selection of actions.

Claim 32 Let $k(i)$ be the number of times LLM used action i . Then $(k(1), \dots, k(N))$ is ℓ balanced, where the makespan of the LLM is in the range $[\ell, \ell + 1]$.

In order to bound the regret, we need to lower bound the performance of the optimal allocation.

Lemma 33 Let $\hat{p}(i)$ be the empirical loss of action i , and $\hat{p}(i) = p(i)(1 + \delta(i))$. Then the optimal makespan is at least $(1 - \delta)T/P$, where $\delta = \max_i |\delta(i)|$. In addition, with probability $1 - N\eta$, we have that $\delta < \sqrt{(\log(1/\eta))/T}$.

Proof: The optimal makespan has value T/\hat{P} where $\hat{P} = \sum_{i=1}^N 1/\hat{p}(i)$. By our assumption $\hat{p}(i) > p(i)(1 - \delta)$, and the lemma follows. ■

We bounded, with high probability, the deviation of the LLM scheduler above the T/P bound (Lemma 31), and the deviation of the optimal below the T/P bound (Lemma 33). Therefore, we derived the following theorem.

Theorem 34 With probability $1 - 3NT\eta$ we have that the regret is at most $O(\sqrt{(\log(1/\eta))/T})$.

The LLM scheduler, as any deterministic scheduler that always assign at each time step all the weight to a single action, is bound to have a regret of the order of at least \sqrt{T} with some constant probability. As stated in the the introduction, for any such scheduler, the sum of the loads is a sum of T IID Bernoulli random variables, and with constant probability some load would be of the order of \sqrt{T} above the expectation. We summarize the result in the following theorem.

Theorem 35 The expected regret of any deterministic scheduler (including LLM) is $\Omega(\sqrt{T})$, for $N = 2$.

The influence of the number of actions N is interesting. Somewhat counterintuitively, the regret may drop with the increase in N . To slightly de-mystify this, note that when $T = N$ the makespan of LLM and also the uniform weights is bounded by 1, and therefore the regret is at most 1.

8 Simulations

We demonstrate the algorithms introduced in this paper by running several toy simulations with makespan cost. Of particular interest in the simulations below is the behaviour of the algorithms in different phases the way their variance changes.

We compare the algorithms we developed to several standard algorithms and show considerable gain for all variants of the generic algorithm. As in the theoretical part of the paper we consider two settings, the case where the distribution D is known and where the distribution D is unknown. In the known distribution model we compare the optimal allocation given the distribution to several applications of our algorithm, namely both anytime and the logarithmic regret. In the unknown distribution we compare our algorithms to the least loaded machine algorithm, and the algorithm presented by [4] that is suited to an adversarial setting. We use our algorithm for unknown distribution, with a small modification where in the first phase instead of using random weights we run least loaded machine algorithm (to avoid random guess in the beginning).

Table 1 summarizes the behavior of the different algorithms that were run 200 times where each run consisted of 10^7 time steps. The table demonstrates a few interesting points. First, all of variants of our algorithm enjoy lower regret, and even more striking they enjoy a very low standard deviation compared to the more intuitive algorithms (LLM and the a-priori optimal). This suggests that one can regard our method as a method for a variance reduction. The first two rows of Table 1 present the case where the distribution D is a product distribution, where each mean value is chosen uniform at random at $[0, 1]$. The last two rows present a case where the distribution D is correlated and in particular is the multinomial distribution.

	G_{∞}^{kn}	G_{∞}^{un}	$G_{\infty}^{kn}(\text{anytime})$	LLM	A-priori Optimal	Adversarial
Uniform D						
Average regret	1.6380	6.502	10.3375	20.3173	75.3860	169.3265
STD regret	1.2586	6.0256	5.5891	31.9719	39.8705	104.1208
Multinomial D						
Average regret	1.8890	5.4982	8.1424	12.0866	47.0461	160.6345
Std regret	1.3544	4.5942	4.0043	20.5329	24.9777	98.2591

Table 1: Regret of different algorithms for uniform and correlated D .

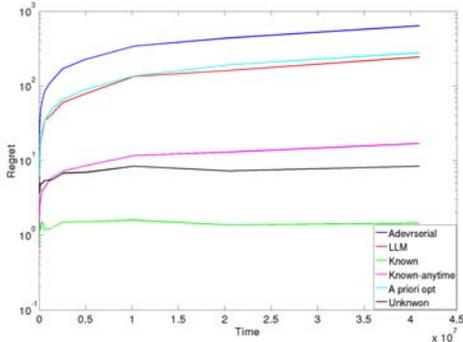


Figure 1: A demonstration of the regret growth of the algorithm when the termination time is known. The number of machines is 8 and each point is an average of 100 runs. In each time D is a product distribution where the expectation of each machine is chosen at uniform at $[0, 1]$.

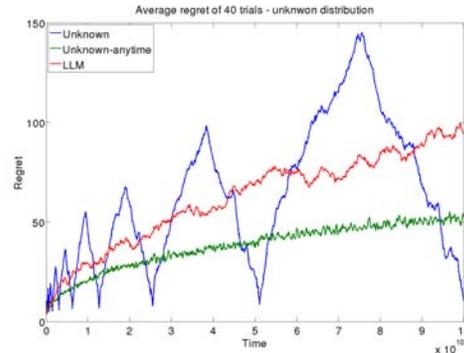


Figure 2: A demonstration of the regret the different algorithms as the time grows. All algorithms know the termination time which is 10^8 and each point is an average of 20 runs.

As can be observed, the results of for correlated and product D are similar demonstrating the ability of our algorithm to work even when there are correlations. Figure 1 depicts the regret of each algorithm as a function of the termination time T .⁴ As expected, the algorithm which balances at the end outperforms the other algorithms significantly. Another point of interest is that although the adversarial and the LLM algorithm have both $O(\sqrt{T})$ regret, the LLM algorithm outperform the adversarial one.

Figure 2 provides some intuition on how the logarithmic regret algorithm G_{∞}^{un} behaves during each phase. Namely, the regret grows at the beginning of a phase and then starts to decrease until it is very small. In addition, it demonstrates the advantage of the anytime algorithm over other algorithms.

References

- [1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.
- [2] P. Auer and R. Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems 19*, pages 49–56. MIT Press, 2007.
- [3] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, 2006.
- [4] E. Even-Dar, R. Kleinberg, S. Mannor, and Y. Mansour. Online learning for global cost functions. In *The 22nd Annual Conference on Learning Theory (COLT) 2009*, pages 41–52, 2009.
- [5] T.L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.
- [6] H. Robbins. Some aspects of sequential design of experiments. *Bull. Amer. Math. Soc.*, 55:527–535, 1952.
- [7] A. Tewari and P. L. Bartlett. Optimistic linear programming gives logarithmic regret for irreducible MDPs. In *Advances in Neural Information Processing Systems Conference (NIPS)*, 2007.

⁴Note that algorithm at time 10000 and the one at time 20000 will be different since they both know the end time.

Hedging Structured Concepts

Wouter M. Koolen*
Advanced Systems Research
Centrum Wiskunde en Informatica
wmkoolen@cwi.nl

Manfred K. Warmuth†
Department of Computer Science
UC Santa Cruz
manfred@cse.ucsc.edu

Jyrki Kivinen‡
Department of Computer Science
University of Helsinki
jkivinen@cs.helsinki.fi

Abstract

We develop an online algorithm called *Component Hedge* for learning structured concept classes when the loss of a structured concept sums over its components. Example classes include paths through a graph (composed of edges) and partial permutations (composed of assignments). The algorithm maintains a parameter vector with one non-negative weight per component, which always lies in the convex hull of the structured concept class. The algorithm predicts by decomposing the current parameter vector into a convex combination of concepts and choosing one of those concepts at random. The parameters are updated by first performing a multiplicative update and then projecting back into the convex hull. We show that Component Hedge has optimal regret bounds for a large variety of structured concept classes.

1 Introduction

We develop online learning algorithms for structured concepts that are composed of components. For example, sets are composed of elements, permutations of individual assignments, trees have edges as components, etc. The number of components d is considered small, but the number of structured concepts D built from the components is typically exponential in d .

Our algorithms address the following online prediction problem. In each trial the algorithm first produces a concept from the structured class by choosing a concept probabilistically based on its current parameters. It then observes the loss of each concept. Finally, it prepares for the next trial by updating its parameters by incorporating the losses. Since the algorithm “hedges” by choosing the structured concept probabilistically, we analyze the expected loss incurred in each trial. The goal is to develop algorithms with small regret, which is the total expected loss of the online algorithm minus the loss of the best structured concept in the class chosen in hindsight.

We now make a key simplifying assumption on the loss: We assume that the loss of a structured concept in each trial is always the sum of the losses of its components and that the component losses always have range $[0, 1]$. Thus if the concepts are k -element sets chosen out of n elements, then in each trial each element is assigned a loss in $[0, 1]$ and the loss of any particular k -set is simply the sum of the losses of its elements. Similarly for trees, a loss in $[0, 1]$ is assigned to each edge of the graph and the loss of a tree is the sum of the losses of its edges.

We will show that with this simplifying assumption we still have rich learning problems that address a variety of new settings. We give efficient algorithms (i.e. polynomial in d) that serve as an entry point for considering more complex losses in the future.

Perhaps the simplest approach to learning structured concept classes online is Follow the Perturbed Leader (FPL) algorithm [KV05]. FPL adds a random perturbation to the cumulative loss of each individual component, and then plays the structured concept with minimal perturbed loss. FPL is widely applicable, since efficient combinatorial optimization algorithms exist for a broad range of concept classes. Unfortunately, the loss range of the structured concepts enters into the regret bounds that we can prove for FPL. For example,

*Supported by BRICKS project AFM2.2. Part of this research was performed while visiting UCSC supported by NSF grant IIS-0917397.

†Supported by NSF grant IIS-0917397

‡Part of this research was performed while visiting UCSC. Supported by Academy of Finland grant 118653 (Algodan) and the PASCAL Network of Excellence.

for k -sets the loss range is $[0, k]$ because each set contains k elements, for permutations the loss range is $[0, n]$ because each permutation is composed of n assignments, etc.

A second simple approach for learning well compared to the best structured concept is to run the Hedge algorithm of [FS97] with one weight per structured concept. The original algorithm was developed for the so-called expert setting, which in the context of this paper corresponds to learning with sets of size one. To apply this algorithm to our setting, the experts are chosen as the structured concepts in the class we are trying to learn. In this paper we call this algorithm *Expanded Hedge* (EH). It maintains its uncertainty as a probability distribution over all structured concepts and the weight W_C of concept C is proportional to $\exp(-\eta\ell(C))$, where $\ell(C)$ is the total loss of concept C incurred so far and η is a non-negative learning rate.

There are two problems with EH. First, there are exponentially many weights to maintain. However our simplifying assumption assures that $\ell(C)$ is a sum over the losses of the component of C . This implies that W_C is proportional to a product over the components of the structured concept C and this fact can be exploited to still achieve efficient algorithms in some cases. More importantly however, like for FPL, the loss range of the structured concepts usually enters into the best regret bounds that we can prove.

Learning with structured concepts has also been dealt with recently in the bandit domain [CBL09]. However all of this work is based on EH and contains the additional range factors.

Our contribution Our new method, called *Component Hedge* (CH), avoids the additional range factors altogether. Each structured concept C is identified with its incidence vector in $\{0, 1\}^d$ indicating which components are used. The parameter space of CH is simply the convex hull of all concepts in the class \mathcal{C} to be learned. Thus, whereas EH maintains a weight for each structured concept, CH only maintains a weight for each component. The current parameter vector represents CH’s first-order “uncertainty” about the quality of each concept. The value of parameter i represents the *usage* of component i in the next prediction. The usages of the components are updated in each trial by incorporating the current losses, and if the usage vector leaves the hull, then it is projected back via a relative entropy projection. The key trick to make this projection efficient is to find a representation of the convex hull of the concepts as a convex polytope with a number of facets that is polynomial in d . We give many applications where this is possible.

We clearly champion the Component Hedge algorithm in this paper because we can prove regret bounds for this algorithm that are tight within constant factors for many structured concept classes. Also it is trivial to enhance CH with a variety of “share updates” that make it robust in the case when the best comparator changes over time [HW98, BW02].

Two instances of CH have appeared before even though this name was not used: learning with k -sets [WK08] and learning with permutations [HW09]. The same polytope we use for paths was also employed in [AHR08] for developing online algorithms for the bandit setting. They avoid the projection step altogether by exploiting a barrier function. The contribution of this paper is to clearly formulate the general methodology of the Component Hedge algorithm and give many more involved combinatorial examples. In the case of permutations we also show how the method can be used to learn truncated permutations. Also in earlier work [TW03] it was pointed out that the Expanded Hedge algorithm can be simulated efficiently in many cases. In particular, the concept class of paths in a directed graph was introduced. However, good bounds were only achieved in very special cases. In this paper we show that CH essentially is optimal for the path problem.

Paper outline We give the basic setup for the structured prediction task, introduce CH and prove its general regret bound in Section 2. We then turn to a list of applications in Section 3: vanilla experts, k -sets, permutations, paths, undirected and directed spanning trees. For each structured concept class we discuss efficient implementation of CH, and derive expected regret bounds for this algorithm. Then in Section 4 we provide matching lower bounds for all examples, showing that the regret of CH is optimal within a constant factor. In Section 5 we compare CH to the existing algorithms EH and FPL. We observe that the best general regret bounds for each algorithm exceed that of CH by a significant range factor. We show that the bounds for these other algorithms can be improved to closely match those of CH whenever the so-called *unit rule* holds for the algorithms and class. This means any loss vector $\ell \in [0, 1]^d$ can be split into up to d scaled unit loss vectors $\ell_i e_i$ and processing these in separate trials always incurs at least as much loss. Unfortunately, for most pairing of the algorithms CH and FPL with the classes we consider in this paper, we have explicit counter examples to the unit rule. Finally, Section 6 concludes with a list of open problems.

2 Component Hedge

Prediction task We consider sequential prediction [HKW98, CBL06] over a structured concept class [KV05, CBL09]. Fix a set of concepts $\mathcal{C} \subseteq \{0, 1\}^d$ of size $D = |\mathcal{C}|$. For example \mathcal{C} could consist of the incidence vectors of subsets of k out of n elements (then $D = \binom{n}{k}$ and $d = n$), or the adjacency matrices of undirected spanning trees on n elements (then $D = (n-1)^{n-2}$ and $d = n(n-1)/2$).

Our online learning protocol proceeds in trials. At trial t , we have to produce a single concept $C^t \in \mathcal{C}$. Then a loss vector $\ell^t \in [0, 1]^d$ is revealed, and we incur loss given by the dot product $C^t \cdot \ell^t$. Although each

Table 1: Example structured concept classes

Case	U	D	d
Experts	1	n	n
k -Sets	k	$\binom{n}{k}$	n
Permutations	n	$n!$	n^2
Paths (from source via $\leq n$ intermediate nodes to sink)	$n + 1$	$n! \cdot e - o(1)$	$n(n + 1) + 1$
Undirected spanning trees	$n - 1$	n^{n-2}	$n(n - 1)/2$
Directed spanning trees w. fixed root	$n - 1$	n^{n-2}	$(n - 1)^2$

component suffers loss at most 1, a concept may suffer loss up to $U := \max_{C \in \mathcal{C}} |C|$. We allow randomized algorithms. Thus the expected loss of the algorithm at trial t is $\mathbb{E}[\mathbf{C}^t] \cdot \ell^t$, where the expectation is over the internal randomization of the algorithm. Our goal is to minimize our (*expected*) *regret* after T trials

$$\sum_{t=1}^T \mathbb{E}[\mathbf{C}^t] \cdot \ell^t - \min_{C \in \mathcal{C}} \sum_{t=1}^T C \cdot \ell^t.$$

That is, the difference between our cumulative expected loss and the loss of the best concept in hindsight.

Note that the i th component of $\mathbb{E}[\mathbf{C}^t]$ is the probability that component i is “used in” concept \mathbf{C}^t . We therefore call $\mathbb{E}[\mathbf{C}^t]$ the *usage vector*. This vector becomes the internal parameter of our algorithm. The set of all usages vector is the convex hull of the concepts.

2.1 Component Hedge

Two instances of CH appeared before in the literature [HW09, WK08]. Here we give the algorithm in its general form, and prove a general regret bound. The algorithm CH maintains its uncertainty about the best structured concept as a usage vector w^t in $\text{conv}(\mathcal{C}) \subseteq [0, 1]^d$, the convex hull of the concepts \mathcal{C} . The initial weight w^0 is typically the usage of the uniform distribution on concepts. CH predicts in trial t by decomposing w^{t-1} into a convex combination¹ of the concepts \mathcal{C} , then sampling \mathbf{C}^t according to its weight in that convex combination. The expected loss of CH is thus $w^{t-1} \cdot \ell^t$. The updated weight w^t is obtained by trading off the relative entropy with the linear loss:

$$w^t := \operatorname{argmin}_{w \in \text{conv}(\mathcal{C})} \Delta(w \| w^{t-1}) + \eta w \cdot \ell^t, \quad \text{where} \quad \Delta(w \| v) = \sum_{i \in [d]} \left(w_i \ln \frac{w_i}{v_i} + v_i - w_i \right).$$

It is easy to see that this update can be split into two steps: an unconstrained update followed by relative entropy projection into the convex hull:

$$\begin{aligned} \hat{w}^t &:= \operatorname{argmin}_{w \in \mathbb{R}^d} \Delta(w \| w^{t-1}) + \eta w \cdot \ell^t \\ w^t &:= \operatorname{argmin}_{w \in \text{conv}(\mathcal{C})} \Delta(w \| \hat{w}^t). \end{aligned}$$

It is easy to see that $\hat{w}_i^t = w_i^{t-1} e^{-\eta \ell_i^t}$, that is, the old weights are simply scaled down by the exponentiated losses. The result of the relative entropy projection w^t unfortunately does not have a closed form expression.

For CH to be efficiently implementable, the hull has to be captured by polynomial in d many constraints. This will allow us to efficiently decompose any point in the hull as a convex combination of at most $d + 1$ concepts. The trickier part is to efficiently implement the projection step. For this purpose one can use generic convex optimization routines. For example this was done in the context of implementing the entropy regularized boosting algorithm [WGV08]. We proceed on a case by case basis and often develop iterative algorithms that locally enforce constraints and do multiple passes over all constraints. See Table 1 for a list of structured concept classes we consider in this paper.

2.2 Regret bounds

As in [HW09], the analysis is split into two steps paralleling the two update steps. Essentially the unnormalized update step already gives the regret bound and the projection step does not hurt. For any usage vector

¹This decomposition usually is far from unique.

$w^{t-1} \in \text{conv}(\mathcal{C})$, loss vector $\ell^t \in \{0, 1\}^d$ and any comparator concept C ,

$$\begin{aligned} (1 - e^{-\eta})w^{t-1} \cdot \ell^t &\leq \underbrace{\Delta(C\|w^{t-1}) - \Delta(C\|\hat{w}^t)}_{\sum_i w_i^{t-1}(1-e^{-\eta\ell_i^t})} + \eta C \cdot \ell^t \\ &\leq \Delta(C\|w^{t-1}) - \Delta(C\|w^t) + \eta C \cdot \ell^t \end{aligned}$$

The first inequality is obtained by bounding the exponential using the inequality $1 - e^{-\eta x} \geq (1 - e^{-\eta})x$ for $x \in [0, 1]$ as done in [LW94]. The second inequality is an application of the Generalized Pythagorean Theorem [HW01], using the fact that w^t is a Bregman projection of \hat{w}^t into the convex set $\text{conv}(\mathcal{C})$, which contains C . We now sum over trials and obtain, abbreviating $\ell^1 + \dots + \ell^T$ to $\ell^{\leq T}$,

$$(1 - e^{-\eta}) \sum_{t=1}^T w^{t-1} \cdot \ell^t \leq \Delta(C\|w^0) - \Delta(C\|w^T) + \eta C \cdot \ell^{\leq T}.$$

Recall that $w^{t-1} \cdot \ell^t$ equals the expected loss $\mathbb{E}[C^t] \cdot \ell^t$ of CH in trial t . Also, relative entropies are nonnegative, so we may drop the second one, giving us the following bound on the total loss of the algorithm:

$$\sum_{t=1}^T \mathbb{E}[C^t] \cdot \ell^t \leq \frac{\Delta(C\|w^0) + \eta C \cdot \ell^{\leq T}}{1 - e^{-\eta}}.$$

To proceed we have to expand the prior w^0 . We consider the *symmetric balanced* case, i.e. where the concept class is invariant under permutation of the components, and every concept uses exactly U components. Paths may have different lengths and hence do not satisfy these requirements. All other examples from Table 1 do. In this balanced symmetric case we take w^0 to be the usage of the uniform distribution on concepts, satisfying $w_i^0 = U/d$ for each component i . It follows that $\Delta(C\|w^0) = U \ln(d/U)$, because any comparator C is a $0/1$ vector that also uses exactly U components.

Let ℓ^* denote $\min_{C \in \mathcal{C}} C \cdot \ell^{\leq T}$, the loss of the best concept in hindsight. Then by choosing $\eta = \sqrt{\frac{2U \ln(d/U)}{\ell^*}}$ as a function of ℓ^* , we obtain the following general expected regret bound for CH:

$$\mathbb{E}[\ell_{\text{CH}}] - \ell^* \leq \sqrt{2\ell^* U \ln(d/U)} + U \ln(d/U). \quad (1)$$

The best-known general regret bounds for Expanded Hedge [FS97] and Follow the Perturbed Leader [HP05] are:

$$\mathbb{E}[\ell_{\text{EH}}] - \ell^* \leq \sqrt{2\ell^* U \ln D} + U \ln D \quad (2)$$

$$\mathbb{E}[\ell_{\text{FPL}}] - \ell^* \leq \sqrt{4\ell^* U d \ln d} + 3U d \ln d \quad (3)$$

where $D = |\mathcal{C}|$. Specific values for U , D and d in each application are listed in Table 1. We remark that if only an upper bound $\hat{\ell} \geq \ell^*$ is available, then we can still tune η as a function of $\hat{\ell}$ to achieve these bounds with $\hat{\ell}$ under the square roots instead of ℓ^* . Moreover, standard heuristics can be used to tune η “online” when no good upper bound on ℓ^* is given, which increase the expected regret bounds by at most a constant factor. (e.g. [CBFH⁺97, HP05]).

We are not concerned with small multiplicative constants (e.g. 2 vs 4), but the gap between (1) and both (2) and (3) is significant. To compare, observe that $\ln D$ is of order $U \ln d$ in all our applications. Thus, the EH regret bound is worse by a factor \sqrt{U} , while FPL is worse by a bigger factor \sqrt{d} . Moreover, in Section 4 we show for the covered examples that our expected regret bound (1) for CH is optimal up to constant scaling.

Some concept classes have special structure that can be exploited to improve the regret bounds of FPL and EH down to that of CH. We consider one such property, called the *unit rule* in Section 5.

3 Applications

We consider the following structured concept classes: experts, k -sets, truncated permutations, source-sink paths, undirected and directed spanning trees. In each case we discuss implementation of CH and obtain a regret bound. Matching lower bounds are presented in Section 4.

3.1 Experts

The most basic example is the vanilla expert setting. In this case, the set of “structured” concepts equals the set of n standard basis vectors in \mathbb{R}^n . We will see that in this case Component Hedge gracefully degrades to the original Hedge algorithm. First, the parameter spaces of both algorithms coincide since the convex

hull of the basis vectors equals the probability simplex. Second, the predictions coincide since a vector in the probability simplex decomposes uniquely into a convex combination of basis vectors. Third, the parameter updates are the same, since the relative entropy projection of a non-negative weight vector into the probability simplex amounts to re-normalizing to unity.

In fact on this simple task CH, EH and FPL each coincide with Hedge. For CH and EH this is obvious. For FPL this fact was observed in [KW05, Kal05] by using log-of-exponential perturbations instead of exponential perturbations used in the original paper [KV05]. Thus, we obtain following regret bound for all algorithms:

$$\mathbb{E}[\ell_{\text{CH}}] - \ell^* \leq \sqrt{2\ell^* \ln n} + \ln n.$$

3.2 k -sets

The problem of learning with sets of k out of n elements was introduced in [WK08] and applied to online Principal Component Analysis (PCA). Their algorithm is an instance of CH, and we review it here. The convex hull of k -sets equals the set of $w \in \mathbb{R}_+^n$ that satisfy the following constraints:

$$w_i \leq 1 \quad \text{for all } i \in [n] \quad \text{and} \quad \sum_{i=1}^n w_i = k. \quad (4)$$

Relative entropy projection into this polytope amounts to re-normalizing the sum to k , followed by redistributing the mass of the components that exceed 1 over the remaining components so that their ratios are preserved. Finally, each element of the convex hull of sets can be greedily decomposed into a convex combination of n k -sets by iteratively removing sets in the convex combination while always setting the coefficient of the new set as high as possible. Both projection and decomposition take $O(n^2)$ time [WK08].

Regret bound By (1), the regret of CH on sets is

$$\mathbb{E}[\ell_{\text{CH}}] - \ell^* \leq \sqrt{2\ell^* k \ln(n/k)} + k \ln(n/k).$$

We give a matching lower bound in Section 4.

3.3 Truncated permutations

The second instantiation of CH that has appeared is the problem of permutations [HW09]. Here we consider a slightly generalized task: *truncated permutations* of k out of n elements. A truncated permutation fills k slots with distinct elements from a pool of n elements. Equivalently, a truncated permutation is a maximal matching in the complete bipartite graph between $[k]$ and $[n]$. Truncated permutations extend k -sets by linearly ordering the selected k elements.

Results to search queries are usually in the form of a truncated permutation; of all n existing documents, only the top k are displayed in order of decreasing relevance. Predicting with truncated permutations is thus a model for learning the best search result.

Matching polytope We write $i \leftarrow j$ for the component that assigns item j to slot i . Now the convex hull of truncated permutations consists of all $w \in \mathbb{R}_+^{k \times n}$ (see [Sch03, Corollary 18.1b]) satisfying the following k row (left) and n column (right) constraints:

$$\sum_{j \in [n]} w_{i \leftarrow j} = 1 \quad \text{for all } i \in [k] \quad \text{and} \quad \sum_{i \in [k]} w_{i \leftarrow j} \leq 1 \quad \text{for all } j \in [n]. \quad (5)$$

Relative entropy projection The relative entropy projection of \hat{w} into the convex hull of truncated permutations $w = \operatorname{argmin}_{w \text{ s.t. (5)}} \Delta(w \| \hat{w})$ has no closed form solution. By convex duality, $w_{i \leftarrow j} = \hat{w}_{i \leftarrow j} e^{-\lambda_i - \mu_j}$, where λ_i and μ_j are the Lagrange multipliers associated to the row and column constraints (5), which minimize

$$\sum_{i \in [k]; j \in [n]} \hat{w}_{i \leftarrow j} e^{-\lambda_i - \mu_j} + \sum_{i \in [k]} \lambda_i + \sum_{j \in [n]} \mu_j.$$

under the constraint that $\mu \geq \mathbf{0}$. This dual problem, which has $2n$ variables and n constraints, may be optimized directly using numerical convex optimization software. Another approach is to iteratively reestablish each violated constraint beginning from $\mu = \mathbf{0}$ and $\lambda = \mathbf{0}$. In full permutation case ($k = n$), this process is called *Sinkhorn balancing*. It is known to converge to the optimum, see [HW09] for an overview of efficiency and convergence results of this iterative method.

Decomposition Our decomposition algorithm for truncated permutations interpolates between the decomposition algorithms used for k -sets and full permutations [WK08, HW09]. Assume w lies in the hull of truncated permutations, i.e. the constraints (5) are satisfied. To measure progress, we define a score $s(w)$ as the number of zero components in w plus the number of column constraints that are satisfied with equality.

Our algorithm maintains a truncated permutation C that satisfies the following invariant: C hits all columns whose constraints are satisfied with equality by w , and avoids all components with weight zero in w . Such a C can be established in time $O(k^2n)$ using augmenting path methods (see [Sch03, Theorem 16.3]).

Let l be the minimum weight of the components used by C , and let h be the maximum column sum of the columns untouched by C . So by construction $h < 1$. If $l = 1$ then $w = C$ and we are done. Otherwise, let $\alpha = \min\{l, 1 - h\}$, and set $w' = (w - \alpha C)/(1 - \alpha)$. It is easy to see that the vector w' satisfies (5), and that $s(w') > s(w)$. It is no longer the case that C satisfies the invariant w.r.t. w' . However, we may compute a weight k matching C' that satisfies the invariant by executing at most $s(w') - s(w)$ many augmenting path computations, which each cost $O(kn)$ time. We describe how this works below. After that we simply recurse on w' and C' . The resulting convex combination is αC plus $(1 - \alpha)$ times the result of the recursion.

The number of iterations is bounded by the score $s(w)$, which is at most kn . Thus, the total running time is $O(k^2n^2)$.

We now show that C can be improved to C' satisfying the invariant by a single augmenting path computation per violated requirement. Let C^* be a size k matching satisfying the invariant for w' . Such a matching always exists because w' lies in the matching polytope. Let $j \in [n]$ be a problematic column, i.e. either C matches j to a row i but $w'_{i \leftarrow j} = 0$, or C does not match j while its column constraint is tight for w' . From j , alternately follow edges from C and C^* . Since C and C^* are both matchings, this can not lead to a cycle, so it must lead to a path. Since all rows are matched, this path must end at a column. The path can not end at a column whose constraint is forced in both C and C^* . So it must end at a column whose constraint is not tight. Incorporating this augmenting path into C corrects the violated requirement without creating any new violations.

Regret bound By (1), the regret of CH on truncated permutations is

$$\mathbb{E}[\ell_{\text{CH}}] - \ell^* \leq \sqrt{2\ell^*k \ln n} + k \ln n.$$

We obtain a matching lower bound in Section 4.

3.4 Paths

The online shortest path problem was considered by [TW03, KV05], and by various researchers in the bandit setting (see e.g. [CBL09, AHR08] and references therein). We develop expected regret bounds for CH for the “full information setting”. Our regret bound improves the bounds given in [TW03, KV05] which have the additional range factors in the square root.

Consider the a directed graph on the set of nodes $[n] \cup \{s, t\}$. Each trial we have to play a walk from the source node s to the sink node t . As always, our loss is given by the sum of the losses of the edges that our walk traverses. Since each edge loss is nonnegative (it lies in $[0, 1]$ by assumption) it is never beneficial to visit a node more than once. Thus w.l.o.g. we restrict attention to paths.

As an example, consider the full directed graph on $[n] \cup \{s, t\}$. Paths of length $k + 1$ through this graph use k distinct internal nodes in order, and therefore are in 1-1 correspondence with truncated permutations of size k . Paths thus generalize truncated permutations by allowing all lengths simultaneously.

Unit flow polytope To implement CH efficiently, we have to succinctly describe the convex hull of paths. Unfortunately, we can not hope to write down linear constraints that capture the convex hull *exactly*. For if we could, then we could solve the *longest path* problem, which is known to be NP complete, by linear programming. Fortunately, there is a slight relaxation of the convex hull of paths that is describable by few constraints, namely the polytope of so-called unit flows. Even better, we will see that this relaxation does not hurt predictive performance at all.

A *unit flow* $w \in \mathbb{R}_+^d$ is described by the following constraints:

$$1 = \sum_{j \in [n]+t} w_{s,j} \quad \text{and} \quad \sum_{j \in [n]+s} w_{j,i} = \sum_{j \in [n]+t} w_{i,j} \quad \text{for each } i \in [n]. \quad (6)$$

We think of $w_{i,j}$ as describing the amount of flow from node i to j . The left constraint ensures that one unit of flow leaves the source s . The right constraint enforces that at internal nodes inflow equals outflow. It easily follows that one unit of flow enters the sink t .

The unit flow polytope is not bounded, but it has the right “bottom”. Namely, the vertices of the unit flow polytope are the s - t paths, see [Sch03, Section 10.3]. The unit flow polytope is the Minkowski sum of the

convex hull of s - t paths and the conic hull (nonnegative linear combinations) of directed cycles. Moreover, each unit flow can be decomposed into at most d paths and cycles, by iterative greedy removal of a directed cycle or paths containing the edge of least non-zero weight in time $O(n^4)$.

Since the unit flow polytope does have polynomially many constraints, we may efficiently run CH on it. Each round, it produces a flow. We then decompose this flow into paths and cycles, and throw away the cycles. We then sample a path from the remaining convex combination of paths.

Relative entropy projection To run CH, we have to compute the relative entropy projection of an arbitrary vector in \mathbb{R}_+^d into the flow polytope (6). This is a convex optimization problem in $d \approx n^2$ variables with constraints. By Slater's constraint condition, we have strong duality. So equivalently, we may solve the concave dual problem, which only has $n + 1$ variables and is unconstrained. The dual problem can therefore be solved efficiently by numerical convex optimization software.

Say we want to find w , the relative entropy projection of \hat{w} into the flow polytope. Since each edge appears in exactly two constraints with opposite sign, the solution has the form $w_{i,j} = \hat{w}_{i,j} e^{\lambda_i - \lambda_j}$ for all $i, j \in [n] \cup \{s, t\}$, where λ_i is the Lagrange multiplier associated with node i (and $\lambda_t = 0$). The vector λ maximizes

$$\lambda_s - \sum_{i \neq t; j \neq s} \hat{w}_{i,j} e^{\lambda_i - \lambda_j}$$

That is, we have to find a single scale factor e^{λ_i} for each node i , such that scaling each edge weight by the ratio of the factors of its nodes reestablishes the flow constraints (6).

We propose the following iterative algorithm. Start with all λ_i equal to zero. Then pick a violated constraint, say at node i , and reestablish it by changing its associated λ_i . That is, we execute either

$$e^{\lambda_s} \leftarrow \frac{1}{\sum_{j \in [n]+t} \hat{w}_{s,j} e^{-\lambda_j}} \quad \text{or} \quad e^{\lambda_i} \leftarrow \sqrt{\frac{\sum_{j \in [n]+s} \hat{w}_{j,i} e^{\lambda_j}}{\sum_{j \in [n]+t} \hat{w}_{i,j} e^{-\lambda_j}}} \quad \text{for some } i \in [n].$$

In our experiments, this algorithm converges quickly. We leave its thorough analysis as an open problem.

Decomposition Find any s - t path with non-zero weights on all edges in time $O(n^2)$. Subtract that path, scaled by its minimum edge weight. This creates a new zero, maintains flow balance, and reduces the source's outflow. After at most n^2 iterations the source has outflow zero. Discard the remaining conic combination of directed cycles. The total running time is $O(n^4)$.

Regret bound for the complete directed graph Since paths have different lengths, we aim for a regret bound that depends on the length of the comparator path. To get such a bound, we need a prior usage vector w^0 that favors shorter paths. To this end, consider the distribution \mathbb{P} that distributes weight 2^{-k} uniformly over all paths of length $k \leq n$, and assigns weight 2^{-n} to the paths of length $n + 1$. This assures that \mathbb{P} is normalized to 1. Since there are $n!/(n - k + 1)!$ paths of length k , the probability of a path P of length k equals

$$\mathbb{P}(\mathbf{P} = P) = \frac{(n - k + 1)!}{2^k n!} \quad \text{if } k \leq n \quad \text{and} \quad \mathbb{P}(\mathbf{P} = P) = \frac{1}{2^n n!} \quad \text{if } k = n + 1.$$

Also, the expected path length $\mathbb{E}[\mathbf{P} \cdot \mathbf{1}]$ is $2 - 2^{-n}$. We now set $w^0 := \mathbb{E}[\mathbf{P}]$, i.e. the usage of \mathbb{P} . There are three kinds of edges. We have one direct edge s, t , we have $2n$ boundary edges of the form s, j or i, t , and we have $n(n - 1)$ internal edges of the type i, j . A simple computation shows that their usages are (for $n \geq 3$)

$$w_{s,t}^0 = \frac{1}{2}, \quad w_{s,j}^0, w_{i,t}^0 = \frac{1}{2n}, \quad w_{i,j}^0 = \frac{1 - 2^{-(n-1)}}{2n(n-1)}.$$

Let P be a comparator path of length k . If $k = 1$ then $\Delta(P \| w^0) = \ln 2$. Otherwise, still for $n \geq 3$,

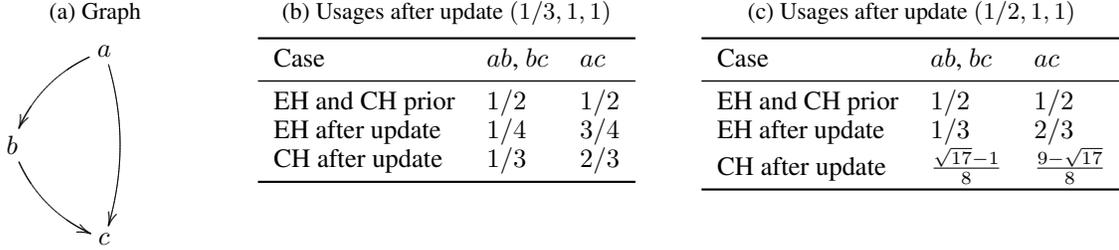
$$\begin{aligned} \Delta(P \| w^0) &= -2 \ln \frac{1}{2n} - (k - 2) \ln \frac{1 - 2^{-(n-1)}}{2n(n-1)} + \mathbb{E}[\mathbf{P} \cdot \mathbf{1}] - k \\ &= (k - 2) \ln(2n(n-1)) + 2 \ln 2n + (k - 2) \ln \left(1 + \frac{2^{-(n-1)}}{1 - 2^{-(n-1)}} \right) - 2^{-n} - (k - 2) \\ &\leq k \ln 2 - (k - 2) \frac{1 - 2^{-n+2}}{1 - 2^{-n+1}} + 2(k - 1) \ln n \leq 2k \ln n. \end{aligned}$$

By tuning η as before, the regret of CH with prior w^0 w.r.t. a comparator path of length k is

$$\mathbb{E}[\ell_{\text{CH}}] - \ell^* \leq \sqrt{4\ell^* k \ln n} + 2k \ln n.$$

This new regret bound improves known results in two ways. First, it does not have the range factors, which in the case of paths usually turn out to be the diameter of the graph, i.e. the length of the longest s - t path. Second, some previous bounds only hold for acyclic graphs. Our bound holds for the complete graph.

Figure 1: EH is not CH on paths



Regret bound for an arbitrary graph We discussed the full graph as a first application of CH. For prediction on an arbitrary graphs we simply design a prior w^0 with zero usage on all edges that are not present in the graph. We could either use graph-specific knowledge, or we could use our old w^0 , disable edges by setting their usage to zero, and project back into the flow polytope. Relative entropy projection never revives zeroed edges. The regret bound now obviously depends on the graph via the prior usage w^0 .

3.4.1 Expanded Hedge and Component Hedge are different on paths

An efficient dynamic programming-based algorithm for EH was presented in [TW03]. This algorithm keeps one weight per edge, just like CH. These weights are updated using the *weight pushing algorithm*. This algorithm performs relative entropy projection on full distributions on paths. Like CH, weight pushing finds a weight of each node, and scales each edge weight by the ratio of its nodes weights. We now show that CH and EH are different on graphs. Consider the graph shown in Figure 1a. Say we use prior \mathbb{P} with weight 1/2 on both paths (a, b, c) and (a, c) . Then the usages are $(1/2, 1/2, 1/2)$ for (ab, bc, ac) . Now multiply edge ab by 1/3 (that is, we give it loss $\ln 3$), and both other edges by 1 (we give them loss zero). The resulting usages of EH and CH are displayed in Table 1b. The usages are different, and hence, so are the expected losses. In most cases (as shown e.g. in Table 1c), the updated usages of CH are irrational while the prior usages and the scale factors of the update are rational. On the other hand, EH always maintains rationality.

3.5 Spanning trees

Whereas paths connect the source to the sink, spanning trees connect every node to every other node. Undirected spanning trees are often used in network-level communication protocols. For example, the *Spanning Tree Protocol* (IEEE 802.1D) is used by mesh networks of Ethernet switches to agree on a single undirected spanning tree, and thus eliminate loops by disabling redundant links. Directed spanning trees are used for asymmetric communication, for example for streaming multimedia from a central server to all connected clients. In either case, the cost of a spanning tree is the sum of the costs of its edges.

Learning spanning trees was pioneered by [KGCC07] for learning dependency parse trees. They discuss efficient methods for parameter estimation under log-loss and hinge loss. [CBL09] derive a regret bound for undirected spanning trees in the bandit setting. We instantiate CH to both directed and undirected trees and give the first regret bound without the range factor.

Three kinds of directed spanning trees are common. Spanning trees with a fixed root, spanning trees with a single arbitrary root, and arborescences (or spanning forests) with multiple roots. We focus on a fixed root. The other two models can be simulated by a fixed root. To simulate arborescences, add a dummy as the fixed root, and put the root selection cost of node i along the path from the dummy to i . Furthermore, to force a single root, increase the cost of all edges leaving the dummy by a fixed huge amount.

Tree polytope To characterize the convex hull of directed trees on n nodes with fixed root 1, we use a trick based on flows from [MW95] that makes use of auxiliary variables $f_{i,j}^k$:

$$0 \leq f_{i,j}^k \leq w_{i,j}, \quad \sum_{i,j} w_{i,j} = n - 1, \quad \underbrace{\sum_{j \neq i} f_{j,i}^k}_{k\text{-flow into } i} + \underbrace{\mathbf{1}_{i=1}}_{k\text{-source at } 1} = \underbrace{\sum_{j \neq i} f_{i,j}^k}_{k\text{-flow out of } i} + \underbrace{\mathbf{1}_{i=k}}_{k\text{-sink at } k}, \quad \text{for } i, j, k \in [n]. \quad (7)$$

The intuition is as follows. A tree has $n - 1$ edges, and every node can be reached from the root. We enforce this by having a separate flow channel f^k for each non-root node k . We place a unit of flow into this channel at the root. Each intermediate node satisfies flow equilibrium. Finally, the target node k consumes the unit of flow destined for it. The first equation ensures that each edge's usage is sufficient for the flow that traverses that edge. The undirected tree polytope is constructed based on the directed tree polytope by considering

the above $w_{i,j}$ as auxiliary variables, an imposing the constraint $w_{i,j} + w_{j,i} = v_{i,j}$. Now v are the weights sought.

Relative entropy projection The relative entropy projection of \widehat{w} into the convex hull of directed spanning trees $w = \operatorname{argmin}_{w \text{ s.t. (7)}} \Delta(w \| \widehat{w})$ has no closed form solution. By convex duality, the solution satisfies

$$w_{i,j} = (n-1) \frac{\widehat{w}_{i,j} e^{\sum_{k \neq 1} \max\{0, \mu_j^k - \mu_i^k\}}}{\sum_{i,j \neq i} \widehat{w}_{i,j} e^{\sum_{k \neq 1} \max\{0, \mu_j^k - \mu_i^k\}}}, \quad f_{ij}^k = \begin{cases} w_{i,j} & \text{if } \mu_j^k > \mu_i^k, \\ 0 & \text{if } \mu_j^k < \mu_i^k, \end{cases}$$

where μ_i^k , the Lagrange multipliers associated to the flow balance constraints, maximize

$$\sum_{k \neq 1} (\mu_k^k - \mu_1^k) - (n-1) \ln \left(\sum_{i,j \neq i} \widehat{w}_{i,j} e^{\sum_{k \neq 1} \max\{0, \mu_j^k - \mu_i^k\}} \right).$$

This unconstrained concave maximization problem in $\approx n^2$ variables seems easier than the primal problem, which has $\approx n^3$ variables and constraints. Note however that the objective is not differentiable everywhere. Alternatively, we may again proceed by iteratively reestablishing constraints locally, starting from some initial assignment to the dual variables μ . This approach is analogous to Sinkhorn balancing.

Decomposition We have no special-purpose tree decomposition algorithm, and therefore resort to a general decomposition algorithm for convex polytopes that is based on linear programming. Let w be in the tree polytope. Choose an arbitrary vertex C (i.e. a spanning tree) by minimizing a linear objective over the current polytope. Now use linear programming to find the furthest point w' in the polytope on the ray from C through w . At least one more inequality constraint is tight for w' . Thus w' lies in a convex polytope of at least one dimension lower. Add this inequality constraint as an equality constraint, recursively decompose w' , and express w as a convex combination of C and the decomposition of w' . The recursion bottoms out at a vertex (i.e. a spanning tree) and the total number of iterations is at most d .

Regret bound By (1), the regret $\mathbb{E}[\ell_{\text{CH}}] - \ell^*$ of CH on undirected and directed spanning trees is at most

$$\sqrt{2\ell^*(n-1) \ln(n/2)} + (n-1) \ln(n/2) \quad \sqrt{2\ell^*(n-1) \ln(n-1)} + (n-1) \ln(n-1)$$

We provide matching lower bounds in Section 4.

4 Lower bounds

Whereas it is easy to get some regret bounds with additional range factors, we show that CH is essentially optimal in all our applications. We leverage the following lower bound for the vanilla expert case:

Theorem 1 *There are positive constants c_1 and c_2 s.t. any online algorithm for q experts with loss range $[0, U]$ can be forced to have expected regret at least*

$$c_1 \sqrt{\ell^* U \ln q} + c_2 \ln q. \quad (8)$$

This type of bound was recently proven in [AWY]. Note that c_1 and c_2 are independent of the number of experts, the range of the losses and the algorithm. Earlier versions of the above lower bound using many quantifier and limit arguments are given in [CBFH⁺97, HW09]. We now prove lower bounds for our structured concept classes by embedding the original expert problem into each class and applying the above theorem. This type of reduction was pioneered in [HW09] for permutations.

The general reduction works as follows. We identify q structured concepts C_1, \dots, C_q in the concept class $\mathcal{C} \subseteq \{0, 1\}^d$ to be learned that partition the d components. Now assume we have an online algorithm for learning class \mathcal{C} . From this we construct an algorithm for learning with q experts with loss range $[0, U]$. Let $\ell \in [0, U]^q$ denote the loss vector for the expert setting. From this we construct a loss vector $L \in [0, 1]^d$ for learning \mathcal{C} : $L := \sum_{i=1}^q \frac{\ell_i}{U} C_i$. That is, we spread the loss of expert i , evenly among the U many components used by concept C_i . Second, we transform the predictions as follows. Say our algorithm for learning \mathcal{C} predicts with any structured concept $C \in \mathcal{C}$. Then we play expert i with probability $C_i \cdot C/U$. The expected loss of the expert algorithm now equals the transformed loss of the algorithm for learning concepts in \mathcal{C} :

$$\mathbb{E}[\ell_i] = \sum_{i=1}^q \frac{C_i \cdot C}{U} \ell_i = C \cdot \sum_{i=1}^q \frac{\ell_i}{U} C_i = C \cdot L$$

This also means that the expected loss of the expert algorithm equals the expected loss of the algorithm for learning the structured class. This implies that the expected regret of the algorithm for learning \mathcal{C} is at least the expected regret of the expert algorithm. The lower bound (8) for the regret in the expert setting is thus also a lower bound for the regret of the structured prediction task.

k -sets We assume that k divides n . Then we can partition $[d]$ with n/k sets, where set i uses components $(i-1)k+1, \dots, ik$. The resulting lower bound has leading factor $\sqrt{k \ln \frac{n}{k}}$, matching the upper bound for CH within constant factors.

Truncated permutations We can partition the n^2 assignments into n full permutations. For example, the n cyclic shifts of the identity permutation achieve this. The truncations to length k of those n permutations partition the kn components in the truncated case. The lower bound with leading factor $\sqrt{k \ln n}$ again matches the regret bound of CH within constant factors.

Spanning trees As observed in [Gus83], the complete undirected graph has $(n-1)/2$ edge-disjoint spanning trees. Hence we get a lower bound with leading factor $\sqrt{(n-1) \ln((n-1)/2)}$. Each undirected spanning tree can be made directed by fixing a root. So there are at least as many disjoint directed spanning trees with a fixed root. In both cases we match the regret of CH within a constant factor.

Paths Consider the directed graph on $[n] \cup s, t$ that has n/k disjoint s - t paths of length $k+1$ connecting source to sink. By construction, we can embed n/k experts with loss range $[0, k]$ into this graph, so the regret has leading factor at least $\sqrt{k \log(n/k)}$. This graph is a subgraph of the complete directed graph $s \rightarrow K_n \rightarrow t$. Moreover, nature can force the algorithm to essentially play on the disjoint path graph by giving all edges outside it sheer infinite loss in a sheer infinite number of trials. This shows that the regret w.r.t. a comparator path of length k through the full graph has leading factor at least $\sqrt{k \log(n/k)}$.

A lower bound on the regret for arbitrary graphs is difficult to obtain since various interesting problems can be encoded as path problems. For example, the expert problem where each expert has a different loss range can be encoded into a graph that has a disjoint path of each length $1, 2, \dots, n$. The optimal algorithm for such expert problems was recently found in [AW], but its regret has no closed form expression. It might be that the regret of CH is tight within constant factors for all graphs, but this question remains open.

5 Comparison to other algorithms

CH is a new member of an existing ecosystem. Other algorithms for structured prediction are EH[LW94] and FPL [KV05]. We now compare them.

Efficiency FPL can be readily applied efficiently to our examples of structured concept classes: k -sets take $O(n)$ per trial using variants of median-finding, truncated permutations take $O(k^2 n)$ per trial using the Hungarian method for minimum weight bipartite matching, paths take $O(n^2)$ per trial using Dijkstra's shortest path algorithm and spanning trees take $O(n^2)$ per trial using either Prim's algorithm or Chu-Liu/Edmonds's algorithm for finding a minimum weight spanning tree.

EH can be efficiently implemented for k -sets [WK08] and paths [TW03] using dynamic programming, and for spanning trees [KGCC07] using the Matrix-Tree Theorem by Kirchoff (undirected) and Tutte (directed). An approximate implementation based on MCMC sampling could be built for permutations based upon [JSV04].

In most cases FPL and EH are faster than CH. This may be partly due to the novelty of CH and the lack of special-purpose algorithms for it. On the other hand, FPL solves a linear minimization problem, which is intuitively simpler than minimizing a convex relative entropy.

5.1 Improved regret bounds with the unit rule

On the other hand, we saw in Section 2.2 that the general regret bound for CH (1) improves the guarantees of EH (1) by a factor \sqrt{U} and those of FPL (3) by a larger factor \sqrt{d} . It is an open question whether these factors are real or simply an artifact of the bounding technique (see Section 6). We now give an example of a property of structured concept classes that makes these range factors vanish.

We say that a prediction algorithm has the *unit rule* on a given structured concept class \mathcal{C} if its worst-case performance is achieved when in each trial only a single expert has nonzero loss. Without changing the prediction algorithm, the unit rule immediately improves its regret bound by reducing the effective loss range of each concept from $[0, U]$ to $[0, 1]$. The improved regret bounds are (c.f. (2) and (3))

$$\mathbb{E}[\ell_{\text{EH}}] \leq \ell^* + \sqrt{2\ell^* \ln D} + \ln D \quad (9)$$

$$\mathbb{E}[\ell_{\text{FPL}}] \leq \ell^* + \sqrt{4\ell^* U \ln d} + 3U \ln d \quad (10)$$

The unit rules for EH and FPL on experts have been observed before [KV05, AWY08]. We reprove them here for completeness. The unit rule holds for both EH and FPL on sets, and for EH on undirected trees. It fails for EH and FPL on permutations, and for EH on directed trees.

We prove the unit rule for EH on sets here, and counter it for EH on directed trees. Proofs and counterexamples for the other cases are similar, and omitted for lack of space.

5.1.1 Unit rule holds for EH on k -sets

Fix an expert i , and let j be an arbitrary other expert. We claim that if we hand out loss to i , then the usage of j increases. For each k -set S , we denote the prior weight of S by W_S . We abbreviate

$$\begin{aligned} Z_i &:= \sum_{S:i \in S} W_S, & Z_{\neg i} &:= \sum_{S:i \notin S} W_S, & Z_j &:= \sum_{S:j \in S} W_S, & Z_{\neg j} &:= \sum_{S:j \notin S} W_S, \\ Z_{i \wedge j} &:= \sum_{S:i \in S, j \in S} W_S, & Z_{\neg i \wedge j} &:= \sum_{S:i \notin S, j \in S} W_S, & Z_{i \wedge \neg j} &:= \sum_{S:i \in S, j \notin S} W_S, & Z_{\neg i \wedge \neg j} &:= \sum_{S:i \notin S, j \notin S} W_S. \end{aligned}$$

Theorem 2 Assume that the prior weights have product structure, i.e. $W_S \propto \prod_{i \in S} w_i$. Then

$$Z_j = \mathbb{P}(j \in \mathbf{S}^1) \leq \mathbb{P}(j \in \mathbf{S}^2 | \ell^1 = \delta_i) = \frac{Z_{i \wedge j} e^{-\eta} + Z_{\neg i \wedge j}}{Z_i e^{-\eta} + Z_{\neg i}}.$$

Proof: With some rewriting, the claim is equivalent to

$$Z_i Z_j \geq Z_{i \wedge j} \quad \text{and also} \quad Z_{i \wedge \neg j} Z_{\neg i \wedge j} \geq Z_{i \wedge j} Z_{\neg i \wedge \neg j}$$

Define

$$R(n, k) := \sum_{\substack{S \subseteq [n] \\ |S|=k}} \prod_{i \in S} w_i.$$

We now show that $R(n, k+1)R(n, m) \geq R(n, k)R(n, m+1)$ for all $0 \leq k < m < n$. The proof proceeds by induction on n . The case $n = 0$ is trivial. Now suppose that the claim holds up to n . We need to show it for $n+1$. For $n > 0$, we have

$$R(n, k) = \mathbf{1}_{k>0} w_n R(n-1, k-1) + \mathbf{1}_{k \leq n} R(n-1, k). \quad (11)$$

Suppose that the induction hypothesis holds up to n . We must show that for all $0 \leq k < m < n+1$

$$R(n+1, k+1)R(n+1, m) \geq R(n+1, k)R(n+1, m+1).$$

By (11), this is equivalent to

$$\begin{aligned} (w_{n+1} R(n, k) + \mathbf{1}_{k < n} R(n, k+1)) (\mathbf{1}_{m > 0} w_{n+1} R(n, m-1) + \mathbf{1}_{m \leq n} R(n, m)) &\geq \\ (\mathbf{1}_{k > 0} w_{n+1} R(n, k-1) + \mathbf{1}_{k \leq n} R(n, k)) (\mathbf{1}_{m+1 > 0} w_{n+1} R(n, m) + \mathbf{1}_{m < n} R(n, m+1)) & \end{aligned}$$

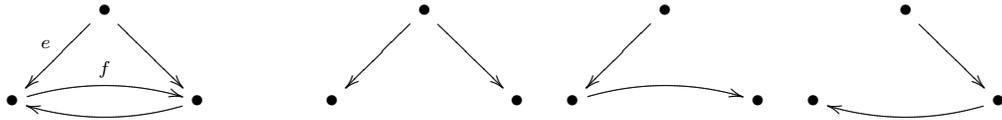
Now we expand, and use $0 \leq k < m < n+1$ to eliminate indicators. It remains to show

$$\begin{pmatrix} (w_{n+1})^2 R(n, k) R(n, m-1) + \\ w_{n+1} R(n, k) R(n, m) + \\ w_{n+1} R(n, k+1) R(n, m-1) + \\ R(n, k+1) R(n, m) \end{pmatrix} \geq \begin{pmatrix} \mathbf{1}_{k > 0} (w_{n+1})^2 R(n, k-1) R(n, m) + \\ \mathbf{1}_{k > 0} \mathbf{1}_{m < n} w_{n+1} R(n, k-1) R(n, m+1) + \\ w_{n+1} R(n, k) R(n, m) + \\ \mathbf{1}_{m < n} R(n, k) R(n, m+1) \end{pmatrix}$$

We now show that this inequality holds line-wise. Lines with active indicators trivially hold. If $k-1 = m$, the second line holds with equality. Otherwise, and for the other lines we use the induction hypothesis. ■

5.1.2 Unit rule fails for EH on directed trees

The unit rule is violated for EH on directed trees. Consider this graph (left) and its three directed spanning trees (right):



Note that we may always restrict attention to a given graph G by assigning zero prior weight to all spanning trees of the full graph that use edges outside G . Now if we put a unit of loss on edge e , the usage of f decreases, and vice versa, contradicting the unit rule. Call the prior weights on directed trees W_A, W_B, W_C . Then the usages satisfy

$$\begin{aligned} W_A + W_B &= \mathbb{P}(e \in \mathbf{T}^1) \geq \mathbb{P}(e \in \mathbf{T}^2 | \ell^1 = \delta_f) = \frac{W_A + W_B e^{-\eta}}{W_A + W_B e^{-\eta} + W_C}, \\ W_B &= \mathbb{P}(f \in \mathbf{T}^1) \geq \mathbb{P}(f \in \mathbf{T}^2 | \ell^1 = \delta_e) = \frac{W_B e^{-\eta}}{W_A e^{-\eta} + W_B e^{-\eta} + W_C}. \end{aligned}$$

6 Conclusion

We developed the Component Hedge algorithm for online prediction over structured expert classes. The advantage of CH is that it has a general regret bound without the range factors that typically plague EH and FPL. We considered several example concept classes, and showed that the lower bound is matched in each case.

Open problems While the unit rule is one method for proving regret bounds for EH and FPL that are close to optimum, there might be other proof methods that show that EH and FPL perform as well as CH when applied to structured concepts. We know of no examples of structured concept classes where EH and FPL are clearly suboptimal. Resolving the question of whether such examples exist is our main open problem.

The prediction task for each structured concept class can be analyzed as a two-player zero-sum game versus nature which tries to maximize the regret. The paper [AWY08] gave an efficient implementation of the minimax optimal algorithm for playing against an adversary in the vanilla expert setting. Actually, the key insight was that the unit rule holds for the optimal algorithm in the vanilla expert case. This fact made it possible to design a balanced algorithm that incurs the same loss no matter which sequence of unit losses is chosen by nature. Unfortunately, the optimum algorithm does not satisfy the unit rule for any of the structured concept classes considered here. However, there might be some sort of relaxation of the unit rule that still leads to an efficient implementation of the optimum algorithm.

In this paper the loss of a structured concept C always had the form $C \cdot \ell$, where ℓ is the loss vector for the components. This allowed us to maintain a mixture of concepts w and predict with a random concept C s.t. $\mathbb{E}[C] = w$. By linearity, the expected loss of such a randomly drawn concept C is the same as the loss of the mixture w . For regression problems with for example the convex loss $(C \cdot \ell - y)^2$ our algorithm can still maintain a mixture w , but now the expected loss of C , i.e. $\mathbb{E}[(C \cdot \ell - y)^2]$, is typically larger than the loss $(w \cdot \ell - y)^2$ of the mixture. We are confident that in this more general setting we can still get good regret bounds compared to the best mixture chosen in hind-sight. All we need to do is replace CH with the more general “Component Exponentiated Gradient” algorithm, which would do an EG update on the parameter vector w and project the updated vector back into the hull of the concepts.

In general, we believe that we have a versatile method of learning with structured concept classes. For example it is easy to augment the updates with a “share update” [HW98, BW02] for the purpose of making them robust against sequences of examples where the best comparator changes over time. We also believe that our methods will get rid of the additional range factors in the bandit setting [CBL09] and that gain versions of the algorithm CH also have good regret bounds.

At the core of our methods lies a relative entropy regularization which results in a multiplicative update on the components. In general, which relative entropy to choose is always one of the deepest questions. For example in the case of learning k -sets, a sum of binary relative entropies over the component can be used that incorporates the $w_i \leq 1$ constraints into the relative entropy term. In general incorporating inequality constraints into the relative entropy seems to have many advantages. However how to do this is an open ended research question.

References

- [AHR08] Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *In Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, 2008.
- [AW] Jacob Abernethy and Manfred K. Warmuth. Repeated games against budgeted adversaries. Unpublished manuscript.
- [AWY] Jake Abernethy, Manfred K. Warmuth, and Joel Yellin. When random play is optimal against an adversary. Journal version of [AWY08], in progress.
- [AWY08] Jacob Abernethy, Manfred K. Warmuth, and Joel Yellin. Optimal strategies for random walks. In *Proceedings of The 21st Annual Conference on Learning Theory*, pages 437–446, July 2008.
- [BW02] Olivier Bousquet and Manfred K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, 2002.
- [CBFH⁺97] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.
- [CBL06] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
- [CBL09] Nicolò Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. In *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.

- [FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.
- [Gus83] Dan Gusfield. Connectivity and edge-disjoint spanning trees. *Information Processing Letters*, 16(2):87–89, 1983.
- [HKW98] David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44(5):1906–1925, 1998.
- [HP05] Marcus Hutter and Jan Poland. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6:639–660, April 2005.
- [HW98] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.
- [HW01] Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.
- [HW09] David P. Helmbold and Manfred K. Warmuth. Learning permutations with exponential weights. *Journal of Machine Learning Research*, 10:1705–1736, July 2009.
- [JSV04] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, 2004.
- [Kal05] Adam Kalai. A perturbation that makes “Follow the Leader” equivalent to “Randomized Weighted Majority”. Private communication, December 2005.
- [KGCC07] Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. Structured prediction models via the Matrix-Tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, 2007.
- [KV05] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.
- [KW05] Dima Kuzmin and Manfred K. Warmuth. Optimum follow the leader algorithm. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT '05)*, pages 684–686. Springer-Verlag, June 2005. Open problem.
- [LW94] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994. Preliminary version appeared in the Proceedings of the 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, 1989.
- [MW95] Thomas L. Magnanti and Laurence A. Wolsey. Optimal trees. In M. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 503–615. North-Holland, 1995.
- [Sch03] Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer-Verlag, Berlin, 2003.
- [TW03] Eiji Takimoto and Manfred K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773–818, 2003.
- [WGV08] Manfred K. Warmuth, Karen Glöcer, and S.V.N. Vishwanathan. Entropy regularized LPBoost. In Yoav Freund, László Györfi, György Turán, and Thomas Zeugmann, editors, *Proceedings of the 19th International Conference on Algorithmic Learning Theory (ALT '08)*, pages 256–271. Springer-Verlag, October 2008.
- [WK08] Manfred K. Warmuth and Dima Kuzmin. Randomized online PCA algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2287–2320, October 2008.

Following the Flattened Leader

Wojciech Kotłowski **Peter Grünwald** **Steven de Rooij**
Centrum Wiskunde & Informatica Centrum Wiskunde & Informatica University of Cambridge
kotlowsk@cwi.nl pdg@cwi.nl steven@statslab.cam.ac.uk

Abstract

We analyze the regret, measured in terms of log loss, of the maximum likelihood (ML) sequential prediction strategy. This “follow the leader” strategy also defines one of the main versions of Minimum Description Length model selection.

We proved in prior work for single parameter exponential family models that (a) in the misspecified case, the redundancy of follow-the-leader is *not* $\frac{1}{2} \log n + O(1)$, as it is for other universal prediction strategies; as such, the strategy also yields suboptimal individual sequence regret and inferior model selection performance; and (b) that in general it is not possible to achieve the optimal redundancy when predictions are constrained to the distributions in the considered model.

Here we describe a simple “flattening” of the sequential ML and related predictors, that does achieve the optimal worst case *individual sequence* regret of $(k/2) \log n + O(1)$ for k parameter exponential family models for bounded outcome spaces; for unbounded spaces, we provide almost-sure results. Simulations show a major improvement of the resulting model selection criterion.

1 Introduction

Let $x_1, x_2, \dots \in \mathcal{X}^*$, be a sequence of outcomes revealed one at a time. After observing $x^n = x_1, x_2, \dots, x_n$, a forecaster assigns a probability distribution on \mathcal{X} , denoted $P(\cdot | x^n)$. Then, after x_{n+1} is revealed, the forecaster incurs the *log loss* $-\log P(x_{n+1} | x^n)$. The performance of the strategy is measured relative to the best in a reference set of strategies, which we call the *model* \mathcal{M} . The difference between the accumulated loss of the prediction strategy and the best strategy in the model is called the *regret*. The goal is to minimize the regret in the worst case over all possible data sequences.

Sequential prediction of individual sequences with log loss has been extensively studied in learning theory, in the framework of *prediction with expert advice* (Azoury & Warmuth, 2001; Cesa-Bianchi & Lugosi, 2001; Cesa-Bianchi & Lugosi, 2006). However, it has also been playing an important role in the information theory: a key result based on the Kraft-McMillan inequality (see, e.g., (Cover & Thomas, 1991)) states that, ignoring rounding issues, for every uniquely decodable codelength function L there is a probability distribution P such that $L(x) = -\log P(x)$ and vice versa¹. Thus, at least when \mathcal{X} is countable, any prediction strategy can also be thought of as a *universal source coding algorithm*; the cumulative logarithmic loss corresponds exactly to the incurred codelength. As Rissanen’s theory of Minimum Description Length (MDL) learning (Barron et al., 1998; Grünwald, 2005) is based on universal coding, a sequential prediction strategy with log loss defines an MDL model selection criterion. Similarly, in statistics Dawid’s theory of prequential model assessment (Dawid, 1984) is based on sequential prediction. Thus we use the terms “prediction strategy” and “code” interchangeably, as we do for “accumulated log loss” and “codelength”.

For parametric models $\mathcal{M} = \{P_\theta | \theta \in \Theta\}$, there are three “universal codes” (prediction strategies with low regret) that are particularly well known in the source coding and MDL communities: (1) after putting a prior distribution π on the model parameters, one can predict using the *Bayesian predictive distribution* $P_{\text{BAYES}}(\cdot | x^n) = \int_{\Theta} P_\theta(\cdot | x^n) \pi(\theta | x^n) d\theta$. (2) If there is a known horizon (maximal number of outcomes), the Shtarkov code (Shtarkov, 1987), also known as the Normalized Maximum Likelihood code (Rissanen, 1996), can be defined. This universal code *minimizes* the worst-case regret. (3) Given an estimator $\hat{\theta} : \mathcal{X}^i \rightarrow$

¹Throughout this text, all logarithms are to the base e and we use nats rather than bits as units of information; however, all results presented here are valid for logarithms of any base.

Θ , one can sequentially select an element of the model using the estimator and use that to predict the next outcome, i.e. $P_{\text{PLUG-IN}}(\cdot | x^n) = P_{\bar{\theta}(x^n)}(\cdot | x^n)$. Such “plug-in codes” were introduced independently in the context of MDL learning (Rissanen, 1984) and in the context of prequential model validation (Dawid, 1984). If we take $\bar{\theta}(x^n)$ equal to the maximum likelihood (ML) estimator $\hat{\theta}(x^n)$, then the resulting strategy is called the “ML plug-in strategy”, which corresponds to the “follow the leader” strategy in learning theory terminology (Kalai & Vempala, 2003; Hutter & Poland, 2005). Strategy (3) always predicts using an element of the model, whereas strategies (1) and (2) do not.

Under weak regularity conditions on the sequence of outcomes, the Bayesian and NML strategies have been shown to achieve asymptotically optimal worst-case regret $(k/2) \log n + O(1)$, where k is the number of parameters of the model (Rissanen, 1989; Rissanen, 1996; Grünwald, 2007). As a consequence, the same $(k/2) \log n + O(1)$ -result holds in expectation and almost surely, if the data are sampled from some distribution P^* , as long as P^* satisfies some very weak regularity conditions. In particular, P^* is *not* required to lie in the model \mathcal{M} : the results still hold if $P^* \notin \mathcal{M}$, i.e. the “model is wrong”, or, as statisticians call it, “the misspecified case”. Now if P^* does lie in \mathcal{M} , then the same $(k/2) \log n + O(1)$ -regret is achieved in expectation under P^* for a large variety of plug-in models including multivariate exponential families, ARMA processes, regression models and so on; examples are (Rissanen, 1986; Hemerly & Davis, 1989; Wei, 1990; Li & Yu, 2000). However, in contrast to the Bayesian and NML results, the plug-in result does *not* hold under misspecification, i.e. if $P^* \notin \mathcal{M}$. We reported earlier (Grünwald & de Rooij, 2005) that under misspecification, already for single parameter exponential family models, the expected regret of the ML plug-in strategy is $\frac{1}{2}c \log n + O(1)$ where c is the variance of an outcome under the true distribution P^* divided by the variance under the element of the model P_θ that minimizes the Kullback-Leibler divergence $D(P^* || P_\theta)$. Moreover, it is shown by (Grünwald & Kotłowski, 2010) that *no* plug-in estimator can achieve $c = 1$ (thus it does not help to replace maximum likelihood predictions by, say, Bayesian posterior mean or moment-estimator-based predictions). This behavior is especially undesirable when the plug-in ML estimator is used to define an MDL or prequential model selection procedure, because in those circumstances, as we explained in Section 6, it is by definition not safe to assume that $P^* \in \mathcal{M}$. This is quite clearly visible in the results of model selection experiments described by (De Rooij & Grünwald, 2006), where the plug-in based version of MDL is significantly outperformed by MDL based on Bayesian and NML strategies.

While the ML plug-in strategy does not achieve the desired expected regret, (Grünwald & Kotłowski, 2010) describe a simple modification of the plug-in prediction strategy that does do so, in the somewhat specific case where $k = 1$ and the outcomes are generated i.i.d. from some distribution P^* . In this paper, we extend this result to the much more general scenario where k can be larger than 1 and where we consider worst-case individual sequence regret rather than expected regret. Our only assumption is that the outcome space \mathcal{X} is bounded in some sense. Following (Grünwald & Kotłowski, 2010), we propose the *flattened* ML prediction strategy, a modification of the ML strategy that puts it slightly outside \mathcal{M} , and in Theorem 11 we show that this strategy achieves optimal asymptotic minimax regret $(k/2) \log n + O(1)$. We also show that when the outcomes are generated i.i.d. from some distribution P^* of which the first four central moments exist, we can remove the assumption of bounded \mathcal{X} and still our prediction strategy achieves the optimal regret $(k/2) \log n$ with probability one.

Our result is important in practice since, in contrast to the Bayesian predictive distribution, the flattened ML strategy is in general just as easy to compute as the ML estimator itself. The flattened ML strategy can be used to define an efficient MDL model selection criterion; we repeated the model selection experiments of (De Rooij & Grünwald, 2006) including this new criterion to find that it displays acceptable performance, unlike the ML plug-in strategy.

Related Work The idea of changing a “follow the leader”-strategy by modifying the leader is not new (Kalai & Vempala, 2003; Hutter & Poland, 2005); however, our “flattened” leader is quite different from the “perturbed” leader described in these earlier papers, and also the setting is quite different: flattened leaders make sense relative to parametric statistical models, which may be regarded as an uncountable set of experts satisfying continuity requirements; perturbed leaders make sense relative to finite or countable sets of otherwise unrelated experts, and the regret bounds obtained in the latter settings are quite different from the $(k/2) \log n$ regret obtained here. The flattened leader is more closely related to the predictive densities considered by (Vidoni, 2008) and (Corcuera & Giummolè, 1999). These authors provide $O(1/n)$ -modifications of the ML density that are similar (but nonequivalent) to ours, and they investigate the behavior of these modifications in terms of expected KL-divergence rather than cumulative regret, in a stochastic, rather than an individual sequence setting.

The paper is organized as follows. We introduce the mathematical context for our results in Section 2. We subsequently define the flattened ML strategy 3 and prove that the regret is $(k/2) \log n + O(1)$ in the individual sequence setting with bounded sample space. We give an example of how this estimator can be used in practice in Section 4, where we apply it to the model of Bernoulli distributions and show how its worst case regret develops as a function of the sample size. In Section 5 we return to theory by providing an

“almost sure” analogue of our individual sequence result, where we can relax the boundedness assumption somewhat. The prediction strategy based on the flattened ML estimator can be used to define an MDL model selection criterion; in Section 6 this criterion is evaluated in a series of model selection experiments, showing that it overcomes many of the weaknesses of the ML plug-in prediction strategy without flattening. We end with a conclusion in Section 7.

2 Notation and Definitions

Let \mathcal{X} be a set of outcomes, taking values either in a finite or countable set, or in a subset of Euclidean space. Exponential family models are families of distributions on \mathcal{X} defined relative to a random variable $\phi : \mathcal{X} \rightarrow \mathbb{R}^k$ (called “sufficient statistic”), and a function $h : \mathcal{X} \rightarrow [0, \infty)$. Let $Z(\eta) = \int_{x \in \mathcal{X}} e^{\eta^T \phi(x)} h(x) dx$ (the integral to be replaced by a sum for countable \mathcal{X}), and $\Theta_{\text{nat}} = \{\eta \in \mathbb{R}^k : Z(\eta) < \infty\}$.

Definition 1 (Exponential family) *The exponential family (Barndorff-Nielsen, 1978) with sufficient statistic ϕ and carrier h is the family of distributions with densities $P_\eta(x) = \frac{1}{Z(\eta)} e^{\eta^T \phi(x)} h(x)$, where $\eta \in \Theta_{\text{nat}}$. Θ_{nat} is called the natural parameter space. The family is called regular if Θ_{nat} is an open and convex subset of \mathbb{R}^k , and if the representation $P_\eta(x)$ is minimal, i.e. the functions $\phi_i(x)$, $i = 1, \dots, k$ are linearly independent.*

We only consider regular exponential families, but this qualification will henceforth be omitted. Examples include the Poisson, geometric and multinomial families, and the model of multidimensional Gaussian distributions. Moreover, without loss of generality, we will make the simplifying assumption that $\phi(x) \equiv x$, i.e. the exponential family is in the canonical form. All results in this paper are valid for more general ϕ .

The statistic $\phi(X) \equiv X$ is sufficient for η (Barndorff-Nielsen, 1978). This suggests reparameterizing the distribution by the expected value of X , which is called the *mean value parameterization*. The function $\mu(\eta) = E_{P_\eta}[X]$ maps parameters in the natural parameterization to the mean value parameterization. It is a diffeomorphism (Barndorff-Nielsen, 1978), therefore the mean value parameter space Θ_{mean} is also an open set of \mathbb{R}^k . We write $\mathcal{M} = \{P_\mu \mid \mu \in \Theta_{\text{mean}}\}$ where P_μ is the distribution with mean value parameter μ .

The sequence of outcomes x_1, \dots, x_n is abbreviated by x^n (x^0 denotes the empty sequence). At every iteration $n = 0, 1, 2, \dots$, the prediction $P(\cdot \mid x^n)$ depends on the past outcomes x^n and has the form of a probability distribution on \mathcal{X} , therefore it can be considered as a conditional of the joint distribution of outcomes in \mathcal{X}^n , which is $P(x^n) = \prod_{i=1}^n P(x_i \mid x^{i-1})$. Conversely, any probability distribution P on the set \mathcal{X}^n defines a prediction strategy induced by its conditional distributions $P(\cdot \mid x^i)$ for $0 \leq i < n$ (Cesa-Bianchi & Lugosi, 2006; Grünwald, 2007).

We are now ready to define the plug-in prediction strategy.

Definition 2 (Plug-in prediction strategy) *Let $\mathcal{M} = \{P_\mu \mid \mu \in \Theta_{\text{mean}}\}$ be an exponential family with mean value parameter domain Θ_{mean} . Given \mathcal{M} , and a function $\bar{\mu} : \mathcal{X}^* \rightarrow \Theta_{\text{mean}}$, define the plug-in prediction strategy $P_{\text{PLUG-IN}}$ by setting, for all n , all x^{n+1} :*

$$P_{\text{PLUG-IN}}(x_{n+1} \mid x^n) = P_{\bar{\mu}(x^n)}(x_{n+1}).$$

We will be mostly concerned with the maximum likelihood (ML) plug-in prediction strategy:

Definition 3 (ML prediction strategy) *Given \mathcal{M} and constants $x_0 \in \Theta_{\text{mean}}$, $n_0 > 0$ we define the ML prediction strategy $P_{\text{ML}}(x_{n+1} \mid x^n)$ as a plug-in strategy with $\bar{\mu} = \hat{\mu}_n^\circ$, where*

$$\hat{\mu}_n^\circ(x^n) = \frac{n_0 x_0 + \sum_{i=1}^n x_i}{n_0 + n}.$$

To understand this definition, note that for exponential families in the mean value parameterization, for any sequence of data, the maximum likelihood parameter $\hat{\mu}_n$ is given by the average $\hat{\mu}_n = n^{-1} \sum x_i$ of the observations (Barndorff-Nielsen, 1978). Here we define our plug-in model in terms of a smoothed maximum likelihood estimator $\hat{\mu}_n^\circ$ that introduces a ‘fake initial outcome’ x_0 with multiplicity n_0 in order to avoid infinite log loss for the first few outcomes, and to ensure that the plug-in ML code of the first outcome is well-defined. The estimator $\hat{\mu}_n^\circ$ can also be interpreted as “maximum a posteriori” estimator, as it maximizes the posterior distribution with appropriate conjugate prior. In practice we can take $n_0 = 1$ but our result holds for any $n_0 > 0$.

Definition 4 (Regret) *We define regret with respect to a sequence x^n of a prediction strategy P relative to the model \mathcal{M} , as a difference between the accumulated log loss of P and the accumulated log loss of the best*

strategy from \mathcal{M} :

$$\begin{aligned}\mathcal{R}(P; x^n) &= \sum_{i=1}^n -\log P(x_i | x^{i-1}) - \inf_{\mu \in \Theta_{\text{mean}}} \sum_{i=1}^n -\log P_{\mu}(x_i) \\ &= -\log P(x^n) - \inf_{\mu \in \Theta_{\text{mean}}} -\log P_{\mu}(x^n).\end{aligned}\tag{1}$$

From the definition, the minimizer:

$$\hat{\mu}_n = \arg \inf_{\mu \in \Theta_{\text{mean}}} -\log P_{\mu}(x^n) = \arg \max_{\mu \in \Theta_{\text{mean}}} P_{\mu}(x^n)$$

is the ordinary maximum likelihood estimator, $\hat{\mu}_n = n^{-1} \sum x_i$. Note, however, that $P_{\hat{\mu}_n}$ is *not* the same as the ML plug-in strategy with $n_0 = 0$: since $P_{\hat{\mu}_n}$ uses the ML estimator *based on the whole sequence* to predict all outcomes from the start, its predictions are generally much better than for the ML plug-in criterion.

Under some mild assumptions about the outcomes, two important prediction strategies, NML (*normalized maximum likelihood*) and Bayes, achieve regrets that are (in an appropriate sense) close to optimal. To be more specific, we must introduce the notion of *ineccsi* subsets of Θ_{mean} and the related sequences (Grünwald, 2007). These are formally defined as follows.

Definition 5 (Ineccsi subsets and sequences) Let \mathcal{M} be a model with a smooth parameterization Θ (e.g., \mathcal{M} may be an exponential family and Θ may represent its mean-value parameterization). The subset $\Theta_0 \subset \Theta$ is *ineccsi* (“interior (is) non-empty; closure (is) compact subset of interior”) if:

1. the interior of Θ_0 is nonempty;
2. the closure of Θ_0 is a compact subset of the interior of Θ .

The sequence x_1, x_2, \dots is a Θ_0 -sequence if there exists m , such that for all $n \geq m$, the ML estimator $\hat{\mu}_n$ exists, is unique and satisfies $\hat{\mu}_n \in \Theta_0$.

Now, the formal definitions of NML and Bayes strategies follow:

Definition 6 (NML prediction strategy) Given \mathcal{M} , an *ineccsi* subset $\Theta_0 \subset \Theta_{\text{mean}}$, and a finite horizon n , define the NML prediction strategy with respect to Θ_0 as:

$$P_{\text{NML}}(x^n) = \frac{\sup_{\mu \in \Theta_0} P_{\mu}(x^n)}{\int_{\mathcal{X}^n} \sup_{\mu \in \Theta_0} P_{\mu}(z^n) dz^n}.$$

Definition 7 (Bayes prediction strategy) Given \mathcal{M} and a probability distribution $\pi(\mu)$ on Θ_{mean} , define the Bayes prediction strategy as:

$$P_{\text{BAYES}}(x^n) = \int_{\Theta_{\text{mean}}} P_{\mu}(x^n) \pi(\mu) d\mu.$$

Note that the NML does not define a random process, since its predictions depend on the horizon n , i.e. marginalizing the NML distribution with some horizon larger than n over the first n outcomes does not yield the NML distribution with horizon n . This is not an issue with the Bayesian strategy, which does define a random process.

The following theorem characterizes the regret of the NML and Bayes prediction strategies:

Theorem 8 Let $\mathcal{M} = \{P_{\mu} \mid \mu \in \Theta_{\text{mean}}\}$ be a k -dimensional exponential family with mean-value parameter space Θ_{mean} . Let Θ_0 be an *ineccsi* subset of Θ_{mean} and let x_1, x_2, \dots be a Θ_0 -sequence. Then,

$$\mathcal{R}(P, x^n) = \frac{k}{2} \log n + O(1),\tag{2}$$

where P is either the NML strategy with respect to Θ_0 with horizon n , or the Bayesian prediction strategy, based on a prior with support Θ_{mean} .

For a proof, see e.g. (Grünwald, 2007). (2) is the famous ‘ k over $2 \log n$ formula’, refinements of which lie at the basis of practical approximations to MDL and Bayesian learning, most notably BIC (Grünwald, 2007). Since the NML strategy in fact *minimizes* the worst-case regret, it follows that a worst-case of $\frac{k}{2} \log n + O(1)$ is optimal. We remark that, if x_1, x_2, \dots do not form an *ineccsi* sequence, then the empirical mean of the x_i tends to the boundary of the parameter space. In that case, the behavior of the Bayesian strategy critically depends on the prior, e.g. with the Bernoulli model and the uniform (Laplace) prior, the worst-case regret becomes $\log n$; with Jeffreys’ prior, it is still $(1/2) \log n + O(1)$ (Freund, 1996); see also Section 4. In this

paper we concentrate on the ineccsi-case, where the data remain bounded away from the boundary, and the $(k/2) \log n$ regret is achieved for Bayes with *all* priors with support Θ_{mean} .

It is known that when outcomes are generated by one of the distributions in \mathcal{M} , the plug-in strategy satisfies (2) as well. However it was shown by (De Rooij & Grünwald, 2005; Grünwald & de Rooij, 2005) that when the outcomes are generated i.i.d. by some distribution P^* outside \mathcal{M} , the ML plug-in strategy P_{ML} behaves suboptimally. Specifically, its expected regret satisfies, for all $\mu^* \in \Theta_{\text{mean}}$,

$$E_{P^*}[\mathcal{R}(P_{\text{ML}}, n)] \geq \frac{1}{2} \frac{\text{var}_{P^*} X}{\text{var}_{P_{\mu^*}} X} \log n + O(1), \quad (3)$$

where $\mu^* = E_{P^*}[X]$ is the element in Θ_{mean} minimizing KL divergence $D(P^* \| P_\mu)$ for $\mu \in \Theta_{\text{mean}}$. A similar result in a different context was already proved earlier by (Wei, 1990). The result was later extended to hold (essentially) for all plug-in prediction strategies (not just ML plug-in) by (Grünwald & Kotłowski, 2010). As (3) is satisfied in the average case, the situation can only become worse in the individual sequence case.

3 The Flattened ML Strategy achieves Optimal Regret

While the plug-in strategies behave suboptimally as shown in the previous section, it remains possible that a small modification of the plug-in strategy, which puts the predictions slightly outside \mathcal{M} , might lead to the optimal regret (2). As a first example, consider the Bayesian predictive distribution when \mathcal{M} is the normal family with fixed variance σ^2 . In this case (see, e.g. (Grünwald, 2007)), the Bayesian code based on prior $\mathcal{N}(\mu_0, \tau_0^2)$ has a simple form $P_{\text{BAYES}}(x_{n+1}|x^n) = f(x_{n+1})$, where f is the density of normal distribution $\mathcal{N}(\mu_n, \tau_n^2)$, with

$$\mu_n = \left(\left(\sum_{i=1}^n x_i \right) + \frac{\sigma^2}{\tau_0^2} \mu_0 \right) / \left(n + \frac{\sigma^2}{\tau_0^2} \right), \quad \text{and} \quad \tau_n^2 = \sigma^2 / \left(n + \frac{\sigma^2}{\tau_0^2} \right).$$

Thus, the Bayesian predictive distribution is itself a Gaussian with mean equal to the smoothed maximum likelihood estimator $\hat{\mu}_n^\circ$ with $n_0 = \sigma^2/\tau_0^2$ and $x_0 = \mu_0$, albeit with a slightly larger variance $\sigma^2 + O(1/n)$. This shows that for the normal family with fixed variance, there exists an “almost” plug-in strategy, which satisfies (2). This led to the conjecture, also in (Grünwald, 2007), that something similar should be possible for exponential families in general. In this section we show that this is indeed the case: we propose a simple modification of the ML strategy, obtained by predicting x_{n+1} using a slightly “flattened” version P_{FML} of the ML strategy P_{ML} , defined as:

Definition 9 (Flattened ML prediction strategy) *Given \mathcal{M} and constants $x_0 \in \Theta_{\text{mean}}$, $n_0 > 0$, we define the flattened ML prediction strategy P_{FML} by setting for all n :*

$$P_{\text{FML}}(x_{n+1}|x^n) := P_{\hat{\mu}_n^\circ}(x_{n+1}) \frac{n + n_0 + \frac{1}{2}(x_{n+1} - \hat{\mu}_n^\circ)^T I(\hat{\mu}_n^\circ)(x_{n+1} - \hat{\mu}_n^\circ)}{n + n_0 + \frac{k}{2}},$$

where $I(\mu)$ is the Fisher information matrix for model \mathcal{M} .

We first check that P_{FML} is properly defined:

Lemma 10 *For every $n = 0, 1, \dots$, $P_{\text{FML}}(x_{n+1}|x^n)$ represents a valid probability distribution, i.e. it is nonnegative and the sum/integral over $x_{n+1} \in \mathcal{X}$ is equal to 1.*

Proof: For every $x \in \mathcal{X}$, $P_{\text{FML}}(x | x^n) \geq 0$ because the information matrix $I(\hat{\mu}_n^\circ)$ is positive definite. To show that $P_{\text{FML}}(\cdot | x^n)$ normalizes to 1, let E_μ denote the expectation with respect to P_μ , i.e. $E_{\hat{\mu}_n^\circ}[f(X)] = \int_{\mathcal{X}} f(x) P_{\hat{\mu}_n^\circ}(x) dx$. Then:

$$\begin{aligned} \int P_{\text{FML}}(x_{n+1}|x^n) dx_{n+1} &= E_{\hat{\mu}_n^\circ} \left[\frac{P_{\text{FML}}(X|x^n)}{P_{\hat{\mu}_n^\circ}(X)} \right] \\ &= \left(n + n_0 + \frac{k}{2} \right)^{-1} \left(n + n_0 + \frac{1}{2} E_{\hat{\mu}_n^\circ} \left[(X - \hat{\mu}_n^\circ)^T I(\hat{\mu}_n^\circ)(X - \hat{\mu}_n^\circ) \right] \right) \\ &= \left(n + n_0 + \frac{k}{2} \right)^{-1} \left(n + n_0 + \frac{1}{2} E_{\hat{\mu}_n^\circ} \left[\text{Tr} \left\{ (X - \hat{\mu}_n^\circ)(X - \hat{\mu}_n^\circ)^T I(\hat{\mu}_n^\circ) \right\} \right] \right) \\ &= \left(n + n_0 + \frac{k}{2} \right)^{-1} \left(n + n_0 + \frac{1}{2} \text{Tr} \left\{ (\text{Cov}_{P_{\hat{\mu}_n^\circ}} X) I(\hat{\mu}_n^\circ) \right\} \right) = 1, \end{aligned}$$

where $\text{Cov}_{P_{\mu_n^0}} X$ is the covariance matrix of $P_{\mu_n^0}$ and the last equality uses a standard result (Barndorff-Nielsen, 1978) for the mean-value parameterization of exponential families which says that for all $\mu \in \Theta_{\text{mean}}$, $\text{Cov}_{\mu} X = I^{-1}(\mu)$. \blacksquare

The predictions of the corresponding flattened ML strategy are not harder to calculate than those of the ordinary ML strategy, which is often much easier than calculating the predictive distribution of the Bayesian strategy. Moreover, we show below in Theorem 11 that under some mild assumptions about the sequence of outcomes, the flattened ML strategy always achieves the optimal regret, satisfying (2). To this end, we need the following two propositions:

Proposition 1 *Let $X \sim P^*$ with mean μ^* , and let \mathcal{M} be the exponential family with sufficient statistic X and mean-value parameter space Θ_{mean} , such that $\mu^* \in \Theta_{\text{mean}}$. Then for every $\mu \in \Theta_{\text{mean}}$ we have:*

$$E_{P^*} [-\log P_{\mu}(X) + \log P_{\mu^*}(X)] = D(\mu^* \|\mu),$$

where $D(\cdot \|\cdot)$ is the Kullback-Leibler divergence.

Proof: By working out both sides of the equation using Definition 1, we find that they both reduce to $\eta(\mu^*)\mu^* - \log Z(\eta(\mu^*)) - \eta(\mu)\mu^* + \log Z(\eta(\mu))$. \blacksquare

Proposition 2 *Let \mathcal{M} be the exponential family with sufficient statistic X and mean-value parameter space Θ_{mean} . Then for every $\mu, \mu^* \in \Theta_0$, where Θ_0 is the inecssi subset of Θ_{mean} , we have:*

$$D(\mu^* \|\mu) = \frac{1}{2}(\mu - \mu^*)^T I(\mu)(\mu - \mu^*) + O(\|\mu - \mu^*\|^3).$$

Proof: We need two standard results regarding the properties of KL divergence (see, e.g. (Barndorff-Nielsen, 1978; Grünwald, 2007)): for any $\mu, \mu^* \in \Theta_{\text{mean}}$, it holds:

1. $D(\mu^* \|\mu) \geq 0$ and the equality only holds for $\mu = \mu^*$,
2. For exponential families, $\partial^2 D(\mu^* \|\mu) / \partial \mu_i \partial \mu_j = I_{ij}(\mu)$.

By Taylor expanding $D(\mu^* \|\mu)$ around μ^* up to the second order, we get:

$$D(\mu^* \|\mu) = D(\mu^* \|\mu^*) + \nabla D(\mu^* \|\mu)^T \Big|_{\mu=\mu^*} (\mu - \mu^*) + \frac{1}{2}(\mu - \mu^*)^T I(\bar{\mu})(\mu - \mu^*),$$

for some $\bar{\mu}$ between μ and μ^* . Due to the first property the zeroth order term disappears; the second order term also disappears because the gradient vanishes at the minimum, so we have:

$$\begin{aligned} D(\mu^* \|\mu) &= \frac{1}{2}(\mu - \mu^*)^T I(\bar{\mu})(\mu - \mu^*) = \frac{1}{2}(\mu - \mu^*)^T I(\mu)(\mu - \mu^*) + \frac{1}{2}(\mu - \mu^*)^T (I(\bar{\mu}) - I(\mu))(\mu - \mu^*) \\ &\leq \frac{1}{2}(\mu - \mu^*)^T I(\mu)(\mu - \mu^*) + \frac{1}{2}\|I(\bar{\mu}) - I(\mu)\| \|\mu - \mu^*\|^2, \end{aligned} \quad (4)$$

where $\|\cdot\|$ denotes vector or matrix norm, depending on the context. Taylor expanding $I(\bar{\mu})$ around μ up to the first order gives $I(\bar{\mu}) = I(\mu) + \nabla I(\tilde{\mu})^T (\bar{\mu} - \mu)$, for some $\tilde{\mu}$ between $\bar{\mu}$ and μ . From that we get:

$$\|I(\bar{\mu}) - I(\mu)\| \leq \|\nabla I(\tilde{\mu})\| \|\bar{\mu} - \mu\| \leq C \|\bar{\mu} - \mu\|, \quad (5)$$

where $C = \sup_{\mu \in \Theta_0} \|\nabla I(\mu)\|$ is finite since closure of Θ_0 is compact and all derivatives of the information matrix are continuous. It follows from the definition of $\bar{\mu}$ that $\|\bar{\mu} - \mu\| \leq \|\mu - \mu^*\|$; using this in (5) and plugging the result into (4) finishes the proof. \blacksquare

Theorem 11 *Let \mathcal{M} be a k -dimensional exponential family with mean-value parameter space Θ_{mean} . Let Θ_0 be an inecssi subset of Θ_{mean} and let x_1, x_2, \dots be a Θ_0 -sequence, i.e. for all $n \geq m$, $\hat{\mu}_n \in \Theta_0$. Moreover, assume that the outcomes are bounded, $\|x_i\| \leq B$ for all $i = 1, 2, \dots$. Then the flattened ML strategy P_{FML} with $x_0 \in \Theta_0$ achieves asymptotically optimal regret, i.e.*

$$\mathcal{R}(P_{\text{FML}}, x^n) = \frac{k}{2} \log n + O(1), \quad (6)$$

where the constant under $O(\cdot)$ depends only on B , Θ_0 and m , while it does not depend on the outcomes x^n .

Proof: Let x_0^n be the sequence of outcomes, composed of n_0 fake outcomes x_0 and the original sequence x^n , i.e. $x_0^n = x_0, \dots, x_0, x_1, \dots, x_n$, and we denote $x_{-i} = x_0, i = 0, \dots, n_0 - 1$. We will use it to cope with the fact that we predict with $\hat{\mu}_n^\circ$ using the ML strategy, while we compare to $\hat{\mu}_n$ in the definition of regret (1). Although $\hat{\mu}_n^\circ$ and $\hat{\mu}_n$ are not the same, they are sufficiently similar that if we replace the term $\log P_{\hat{\mu}_n}(x^n)$ with the term $\log P_{\hat{\mu}_n^\circ}(x_0^n)$ in the definition (1); the difference is only small. Let us denote such a modified regret by $\mathcal{R}'(P_{\text{FML}}, x^{n+1})$. We have:

$$\begin{aligned} \mathcal{R}'(P_{\text{FML}}, x^n) - \mathcal{R}(P_{\text{FML}}, x^n) &= \sum_{i=-n_0-1}^n -\log P_{\hat{\mu}_n^\circ}(x_i) - \sum_{i=1}^n -\log P_{\hat{\mu}_n}(x_i) \\ &= -n_0 \log P_{\hat{\mu}_n^\circ}(x_0) + \sum_{i=1}^n \log \frac{P_{\hat{\mu}_n}(x_i)}{P_{\hat{\mu}_n^\circ}(x_i)} = O(1) + nE_{P_{\text{emp}}} \left[\log \frac{P_{\hat{\mu}_n}(X)}{P_{\hat{\mu}_n^\circ}(X)} \right] \\ &= O(1) + nD(\hat{\mu}_n \parallel \hat{\mu}_n^\circ) = O(1) - \frac{n}{2} (\hat{\mu}_n - \hat{\mu}_n^\circ)^T I(\hat{\mu}_n^\circ) (\hat{\mu}_n - \hat{\mu}_n^\circ) + nO(\|\hat{\mu}_n - \hat{\mu}_n^\circ\|^3), \end{aligned}$$

where P_{emp} is the empirical distribution function, which puts mass $1/n$ on every outcome of x^n , $E_{P_{\text{emp}}}[X] = \hat{\mu}_n$, and we used Proposition 1 with $P^* \equiv P_{\text{emp}}$, and then Proposition 2. Using the fact that:

$$\|\hat{\mu}_n - \hat{\mu}_n^\circ\| = \frac{n_0 \|(x_0 - \hat{\mu}_n)\|}{n + n_0} \leq \frac{2n_0 B}{n},$$

and since $\hat{\mu}_n^\circ \in \Theta_0$ for $n \geq m$, we get for all $n \geq m$:

$$\frac{n}{2} (\hat{\mu}_n - \hat{\mu}_n^\circ)^T I(\hat{\mu}_n^\circ) (\hat{\mu}_n - \hat{\mu}_n^\circ) \leq \frac{n}{2} \|I(\hat{\mu}_n^\circ)\| \|\hat{\mu}_n - \hat{\mu}_n^\circ\|^2 \leq \frac{4n_0^2 B^2}{2n} \sup_{\mu \in \Theta_0} \|I(\mu)\| = O(n^{-1}),$$

where $\|I(\mu)\|$ denotes the matrix norm and we used the fact that $\sup_{\mu \in \Theta_0} \|I(\mu)\|$ is finite due to compactness of the closure of Θ_0 and continuity of information matrix.

Thus, we proved that $\mathcal{R}'(P_{\text{FML}}, x^n) - \mathcal{R}(P_{\text{FML}}, x^n) = O(1)$. To show (6), it now suffices to show that $\Delta(n) = \mathcal{R}'(P_{\text{FML}}, x^{n+1}) - \mathcal{R}'(P_{\text{FML}}, x^n) = \frac{k}{2n} + O(n^{-2})$, where the constant under $O(\cdot)$ does not depend on the outcomes x^n . Then, since $\log n \leq \sum_{i=1}^n \frac{1}{i} \leq \log n + 1$, and $\sum_n n^{-2}$ converges, (6) follows. From the definition, we have:

$$\begin{aligned} \Delta(n) &= -\log P_{\text{FML}}(x_{n+1}|x^n) - \sum_{i=-n_0+1}^{n+1} -\log P_{\hat{\mu}_{n+1}^\circ}(x_i) + \sum_{i=-n_0+1}^n -\log P_{\hat{\mu}_n^\circ}(x_i) \\ &= \log \left(1 + \frac{k}{2(n+n_0)} \right) - \log \left(1 + \frac{1}{2(n+n_0)} (x_{n+1} - \hat{\mu}_n^\circ)^T I(\hat{\mu}_n^\circ) (x_{n+1} - \hat{\mu}_n^\circ) \right) + \sum_{i=-n_0+1}^{n+1} \log \frac{P_{\hat{\mu}_{n+1}^\circ}(x_i)}{P_{\hat{\mu}_n^\circ}(x_i)}. \end{aligned}$$

Let us denote $\xi_n = \frac{1}{2(n+n_0)} (x_{n+1} - \hat{\mu}_n^\circ)^T I(\hat{\mu}_n^\circ) (x_{n+1} - \hat{\mu}_n^\circ)$. We will show that $\log(1 + \xi_n) = \xi_n + O(n^{-2})$.

To this end, we use the fact that for every $z > -1$ it holds $-z \leq -\log(1+z) \leq -z + \frac{z^2}{2}$ (this follows e.g. from a Taylor expansion of $\log(1+z)$ around $z=0$) and show that $\xi_n^2 = O(n^{-2})$:

$$\xi_n^2 \leq \frac{1}{4n^2} \|I(\hat{\mu}_n^\circ)\|^2 \|x_{n+1} - \hat{\mu}_n^\circ\|^4 \leq \frac{1}{4n^2} \left(\sup_{\mu \in \Theta_0} \|I(\mu)\| \right)^2 (2n_0 B)^4 = O(n^{-2}),$$

for all $n \geq m$. Thus we proved $\log(1 + \xi_n) = \xi_n + O(n^{-2})$. Moreover, $\log(1 + \frac{k}{2n}) = \frac{k}{2n} + O(n^{-2})$, so

$$\Delta(n) = \frac{k}{2n} - \frac{1}{2n} (x_{n+1} - \hat{\mu}_n^\circ)^T I(\hat{\mu}_n^\circ) (x_{n+1} - \hat{\mu}_n^\circ) + \sum_{i=-n_0+1}^{n+1} \log \frac{P_{\hat{\mu}_{n+1}^\circ}(x_i)}{P_{\hat{\mu}_n^\circ}(x_i)} + O(n^{-2}). \quad (7)$$

To bound the sum, we note that it equals $(n+n_0+1)D(\hat{\mu}_{n+1}^\circ \parallel \hat{\mu}_n^\circ)$ where we used Proposition 1 with the empirical distribution again. Then, using Proposition 2, we get:

$$\sum_{i=-n_0+1}^{n+1} \log \frac{P_{\hat{\mu}_{n+1}^\circ}(x_i)}{P_{\hat{\mu}_n^\circ}(x_i)} = \frac{n+n_0+1}{2} (\hat{\mu}_n^\circ - \hat{\mu}_{n+1}^\circ)^T I(\hat{\mu}_n^\circ) (\hat{\mu}_n^\circ - \hat{\mu}_{n+1}^\circ) + (n+n_0+1)O(\|\hat{\mu}_n^\circ - \hat{\mu}_{n+1}^\circ\|^3).$$

Since $\hat{\mu}_n^\circ - \hat{\mu}_{n+1}^\circ = (\hat{\mu}_n^\circ - x_{n+1})/(n+n_0+1)$, and $\|\hat{\mu}_n^\circ - x_{n+1}\| \leq 2B$, it follows that:

$$\sum_{i=-n_0+1}^{n+1} \log(P_{\hat{\mu}_{n+1}^\circ}(x_i)/P_{\hat{\mu}_n^\circ}(x_i)) = \frac{1}{2n} (x_{n+1} - \hat{\mu}_n^\circ)^T I(\hat{\mu}_n^\circ) (x_{n+1} - \hat{\mu}_n^\circ) + O(n^{-2}).$$

Putting this into (7) gives: $\Delta(n) = \frac{k}{2n} + O(n^{-2})$, as claimed.

The constant in $O(1)$ does not depend on the sequence x^n , because for $n < m$, $\hat{\mu}_n^\circ$ (as a convex combination of x_0 and $\hat{\mu}_n$) is kept away from the boundary of Θ_{mean} and thus $I(\hat{\mu}_n^\circ)$ is bounded from above by a constant independent of the sequence x^n . \blacksquare

4 Example: the Bernoulli model

The Bernoulli model is $\{P_\mu \mid \mu \in [0, 1]\}$, where $\mathcal{X} = \{0, 1\}$ and $P_\mu(x) = \mu^x(1 - \mu)^{1-x}$. The Fisher information is $I(\mu) = E_{P_\mu}[(\frac{d}{d\mu} \log P_\mu(X))^2] = 1/(\mu(1 - \mu))$. After observing x^n , the likelihood is maximized by $\hat{\mu} = o/n$ where $o = x_1 + \dots + x_n$; we will also use $z = n - o$. It turns out not to be necessary to introduce any fake outcomes in this case (i.e. $n_0 \rightarrow 0$). Thus, $\hat{\mu}_n = \hat{\mu}_n^o$, and the flattened ML prediction is

$$P_{\text{FML}}(1 \mid x^n) = \hat{\mu}_n \left(\frac{n + \frac{1}{2}I(\hat{\mu}_n)(1 - \hat{\mu}_n)^2}{n + \frac{1}{2}} \right) = \frac{n\hat{\mu}_n + \frac{1}{2}(1 - \hat{\mu}_n)}{n + \frac{1}{2}} = \frac{o + \frac{z}{2n}}{n + \frac{1}{2}}.$$

The regret for this estimator is maximized for the all-zero or all-one sequence; an easy calculation shows it to be $-\frac{1}{2} \log(16/\pi) + \log(\Gamma(n + \frac{1}{2})/\Gamma(n)) = \frac{1}{2} \log n + O(1)$. Thus, even though the worst case is achieved for non-inecsi sequences for which technically Theorem 11 does not apply, we find that the flattened ML prediction strategy achieves asymptotically optimal worst-case regret.

In Figure 1, we plot this worst-case regret together with the worst-case regret for a number of other estimators: (1) the traditional Laplace estimator $P(1 \mid x^n) = (o + 1)/(n + 2)$, which is equal to the Bayes predictive distribution using a uniform prior on μ and which does not behave very well on non-inecsi sequences, (2) the Krichevsky-Trofimov estimator $P(1 \mid x^n) = (o + \frac{1}{2})/(n + 1)$ (Krichevsky & Trofimov, 1981), which is equal to the Bayes predictive distribution using Jeffreys' prior, and (3) the ‘‘Last Step Minimax’’ estimator (Takimoto & Warmuth, 2000), also known as ‘‘Conditional NML’’ estimator (Rissanen & Roos, 2007). The regret for this last estimator was shown to be at most $\frac{1}{2} \log(n + 1) + \frac{1}{2}$ in (Takimoto & Warmuth, 2000). As baselines, we plot the functions $\frac{1}{2} \log n$ and $\log n$, as well as the regret under the Shtarkov (or NML) distribution. As mentioned in the introduction, the NML distribution is defined only with respect to a known horizon; here the horizon is increased with the sample size, so the Shtarkov results do not reflect a valid prediction strategy but rather provide a tight lower bound on the worst-case regret.

The figure shows that the flattened ML model shows performance comparable to the KT and last step minimax estimators, although the constant term is slightly higher.

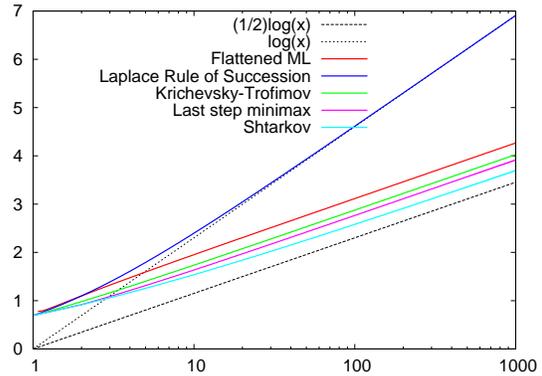


Figure 1: Worst-case regret of Bernoulli estimators.

5 Almost Sure Convergence in the Stochastic Case

In Section 3, we showed that the flattened ML strategy achieves optimal regret under a mild assumption that the outcomes are bounded and form a Θ_0 -sequence. For those cases where this condition is not satisfied, the boundedness requirement can be replaced with the assumption that the data are generated i.i.d. from some distribution of which the first four moments exist. We can then obtain the result (6) with probability one.

The idea of the proof is that when the outcomes are generated i.i.d., they are in some sense bounded with high probability anyway. Specifically, if we allow the bound to increase with n , and if the rate of increase is *faster* than $n^{1/4}$, one can show that when the first four moments of the distribution exist, the outcomes are actually ‘‘bounded’’ (i.e., for all n , the outcome X_n is bounded by the bound for sample size n) with probability one; this is the content of Lemma 12. In Theorem 14, we show that if the bound increases *more slowly* than $n^{1/3}$, most of the analysis done in the proof of Theorem 11 still works. Combining those two facts gives (6) with probability one.

Lemma 12 *Let X_1, X_2, \dots be i.i.d. random variables and suppose the first m moments of X_n exist. Then, for every $B > 0, \alpha > 0$, with probability 1 there exists an n' such that $\|X_n\| \leq Bn^{\frac{1}{m-\alpha}}$ for all $n \geq n'$.*

Proof: Equivalently, we prove that almost surely, the event $A_n = \{\|X_n\| > Bn^{\frac{1}{m-\alpha}}\}$ occurs only finitely often. From the Borel-Cantelli lemma we know that this is the case when $\sum_{n=1}^{\infty} P(A_n) < \infty$; we have

$$P(A_n) = P\left(\|X_n\| \geq Bn^{\frac{1}{m-\alpha}}\right) = P\left(\|X_n\|^m \geq B^m n^{\frac{m}{m-\alpha}}\right) \leq E\|X_n\|^m B^{-m} n^{-\frac{m}{m-\alpha}},$$

where the last step follows from Markov's inequality. Since $\frac{m}{m-\alpha} > 1$, the sum converges. ■

The next lemma is purely technical and will be needed in further proofs:

Lemma 13 Let a_1, a_2, \dots be a positive infinite sequence and let b_1, b_2, \dots be a positive nonincreasing infinite sequence. Define $A_n = a_1 + \dots + a_n$, $B_n = b_1 + \dots + b_n$ and $C_n = a_1 b_1 + \dots + a_n b_n$. If $A_n = O(n)$ and $B_n = O(1)$, then $C_n = O(1)$.

Proof: By assumption there are c, n_0 such that for every $n \geq n_0$ we have $A_n \leq c \cdot n$. Fix some $n \geq n_0$ and A_n . Now suppose there is an $a_{n'} > c$ for some $n_0 \leq n' \leq n$. Since $A_{n'} \leq c \cdot n$, there must be an earlier term $a_{n''} < c$ for some $1 \leq n'' < n'$. By increasing $a_{n''}$ to c and decreasing $a_{n'}$ by the same amount, A_n is unchanged while the value of C_n cannot decrease. Thus we may assume w.l.o.g. that $a_i \leq c$ for all $n_0 \leq i \leq n$. But in that case we have $C_n = C_{n_0-1} + \sum_{i=n_0}^n a_i b_i \leq C_{n_0-1} + c \cdot \sum_{i=n_0}^n b_i = O(1)$. ■

Theorem 14 Let X_1, X_2, \dots be i.i.d. generated by a probability distribution P^* of which the first four moments exist and such that $E[X] \in \Theta_{\text{mean}}$. Let \mathcal{M} be a k -dimensional exponential family with mean-value parameter space Θ_{mean} . Then the flattened ML strategy P_{FML} almost surely achieves asymptotically optimal regret, i.e.

$$\mathcal{R}(P_{\text{FML}}, x^n) = \frac{k}{2} \log n + O(1) \quad (8)$$

holds with probability one.

Proof: Since the first four moments of P^* exists, Lemma 12 states that for large n , the sequence of outcomes x_1, x_2, \dots is bounded by Bn^q for every $q > 1/4$ with probability one. For simplicity, we take $q = 0.3$, but any $q \in (1/4, 1/3)$ would work. From the strong law of large numbers, we know that the smoothed ML estimator $\hat{\mu}_n^\circ$ converges with probability one. Therefore, for large n , $\hat{\mu}_n^\circ$ is bounded, $\|\hat{\mu}_n^\circ\| \leq C$.

We only give the sketch of the proof, because it closely follows the proof of Theorem 11. The main difference is that in Theorem 11, we had $\|x_n\| \leq B$, while here we have (with probability one) $\|x_n\| \leq Bn^{0.3}$ for large n . A closer look at the proof of Theorem 11 shows that after weakening the bound on $\|x_n\|$ we still get the same rates. The only problem is that now we are not able to prove, that $\Delta(n) = \frac{k}{2n} + O(n^{-2})$. However, to obtain (8), it is enough to show that $\Delta(n) = \frac{k}{2n} + f(n)$, where $f(n)$ is a function such that $\sum_n f(n)$ converges and thus is $O(1)$. To this end, instead of directly bounding ξ_n^2 , we will show that $\sum_n \xi_n^2$ converges. Since for large n ,

$$\xi_n^2 \leq \frac{1}{4n^2} \sup_{\|\mu\| \leq C} \|I(\mu)\|^2 (\|x_{n+1}\| + C)^4 = C' \frac{\|x_{n+1}\|^4 + O(n^{0.9})}{n^2}$$

for some constant C' , we only need to show that the sum $\sum_n \frac{\|x_{n+1}\|^4}{n^2}$ converges. But this follows from Lemma 13 with $a_i = \|x_{i+1}\|^4$ and $b_i = i^{-2}$: we have $A_n = \sum_{i=1}^n \|x_{i+1}\|^4 = O(n)$ because A_n/n converges with probability one from the strong law of large numbers (because the fourth moment of P^* exists), and $B_n = \sum_{i=1}^n i^{-2} = O(1)$. This means that with probability one, $\sum_n \xi_n^2$ converges. ■

6 Application: Model Selection

The strange behavior of the ML plug-in code first became apparent in a simulation study, where it was found that this code gives rise to much weaker model selection performance than other model selection criteria, such as Bayes factors model selection or even naive maximum likelihood model selection (De Rooij & Grünwald, 2005; De Rooij & Grünwald, 2006); this is especially disturbing since the plug-in based version of MDL has often been advocated for practical use (Rissanen, 1986; Rissanen, 1989; Grünwald, 2007). As mentioned in the introduction, while the expected regret for the ML plug-in estimator is $\frac{k}{2} \log n + O(1)$ when the model contains the data generating distribution, it behaves differently when it does not. This is quite undesirable for model selection: if it is certain that the true distribution is in all considered models, then there is no need to do model selection in the first place!

Since the flattened ML prequential code described in this paper does not suffer from anomalous redundancy under misspecification, we may reasonably hope for better model selection performance. Therefore we have come full circle by turning back to our original (2005) model selection experiments, in order to determine to what extent the flattened ML prequential plug-in code avoids the shortcomings of the unflattened version, and whether or not it yields a useful model selection criterion.

The experimental setup is the same as it was in (De Rooij & Grünwald, 2006), but we provide a brief description here as well to make this paper self-contained. The experiments involve a number of model selection criteria: one based on the flattened ML plug-in code and a number of others, which will be used as a basis for comparison. After defining these model selection criteria, we show the results of the simulation and discuss how the performance of the criterion based on the flattened ML plug-in estimator relates to the results we reported earlier.

6.1 Experiments

All experiments are based on repeatedly sampling a number of outcomes from either the Poisson model $\mathcal{M}_P = \{P_P(X; \mu) \mid \mu \in (0, \infty)\}$ where $P_P(x; \mu) = e^{-\mu} \mu^x / x!$ or the geometric model $\mathcal{M}_G = \{P_G(X; \mu) \mid \mu \in (0, \infty)\}$ where $P_G(x; \mu) = \mu^x (\mu + 1)^{-(x+1)}$. To make the models easier to compare, both are parameterized by the mean, which is standard for Poisson but not for the geometric model. We first define a number of criteria to select between these models. All criteria can be described in terms of a function L that maps a model \mathcal{M} and a sequence of outcomes x^n to a codelength (negative loglikelihood, accumulated prediction error). Subsequently define the level of evidence in favor of the Poisson model as $\Delta(x^n) = L(\mathcal{M}_G, x^n) - L(\mathcal{M}_P, x^n)$. We select Poisson if $\Delta(x^n) > 0$ and geometric otherwise.

Many common model selection criteria can be defined in terms of a function L . Our experiments involve the following model selection criteria:

The Known mean criterion is defined by $L(\mathcal{M}, x^n) = -\log P(x^n)$; here $P \in \mathcal{M}$ is the distribution that satisfies $E_P[X] = \mu$, where μ is the true mean of the data. Although the true mean is not known in practice, this criterion is useful as an ideal baseline. It has the properties that (1) one of the two hypotheses equals the generating distribution and (2) the sample consists of outcomes which are i.i.d. according to this distribution. In (Cover & Thomas, 1991), Sanov’s Theorem is used to show that in such a situation, the probability that the criterion prefers the wrong model (“error probability”) decreases exponentially in the sample size. If the data are generated using $\text{Poisson}[\mu]$ then the error probability decreases exponentially in the sample size, with some error exponent; if the data are generated with $\text{Geometric}[\mu]$ then the overall probability is exponentially decreasing with the same exponent (Cover & Thomas, 1991, Theorem 12.9.1 on page 312 and text thereafter). Thus, when the error probability is plotted on a log scale, the slope should be equal whether the generating distribution is Poisson or geometric. This can be observed to be the case in Figures 2a and 2b.

The Maximum Likelihood (ML) criterion is defined by $L(\mathcal{M}, x^n) = -\log \sup_{P \in \mathcal{M}} P(x^n)$. This is the same as a (generalized) likelihood ratio test (GLRT) with a threshold of one. The ML criterion is well known to be prone to overfitting: in a complex model, there may be a distribution that provides good fit to the data purely by chance. Two approaches to penalize complex models are known as AIC (Akaike, 1974) and BIC (Schwarz, 1978). However, for both these methods the penalty term depends only on the number of parameters in the models. In this case, both models have only a single parameter, so in $\Delta(x^n)$ the penalty terms cancel: in this case, both AIC and BIC are equivalent to a GLRT with zero threshold!

Bayes factor model selection is obtained if we set $L(\mathcal{M}, x^n) = -\log \int_{\mu} P_{\mu}(x^n) \pi(\mu) d\mu$, where the prior π may depend on the model. In this case, $\Delta(x^n)$ is equal to the logarithm of the Bayes factor. We use Jeffreys’ prior in our experiments. Because it is improper for the Poisson and geometric models, we use the first observation to normalize the prior. Letting $S = \sum_{i=1}^n x_i$, We obtain the following expressions:

$$\begin{aligned} \pi_P(\mu|x_1) &= \frac{e^{-\mu} \mu^{x_1 - \frac{1}{2}}}{\Gamma(\frac{1}{2} + x_1)}; & \pi_G(\mu|x_1) &= (x_1 + \frac{1}{2}) \frac{\mu^{x_1 - \frac{1}{2}}}{(\mu + 1)^{x_1 + \frac{3}{2}}}; \\ L(\mathcal{M}_P, x^n) &= -\log \int_0^{\infty} P_P(x_2^n; \mu) \pi_P(\mu|x_1) d\mu = \log \frac{\Gamma(x_1 + \frac{1}{2})}{\Gamma(S + \frac{1}{2})} + (S + \frac{1}{2}) \log n + \sum_{i=2}^n \log(x_i!); \\ L(\mathcal{M}_G, x^n) &= -\log \int_0^{\infty} P_G(x_2^n; \mu) \pi_G(\mu|x_1) d\mu = -\log(x_1 + \frac{1}{2}) + \log \frac{\Gamma(S + n + \frac{1}{2})}{\Gamma(n) \Gamma(S + \frac{1}{2})}. \end{aligned}$$

The ML plug-in criterion is defined by setting $L(\mathcal{M}, x^n) = -\log P_{ML}(x^n)$ where P_{ML} is as in Definition 3. This codelength does not correspond to a Bayesian marginal likelihood, so this criterion does not yield Bayes factor model selection; however P_{ML} is a valid universal code so it does lead to an MDL model selection procedure.

The flattened ML plug-in criterion is defined by setting $L(\mathcal{M}, x^n) = -\log P_{FML}(x^n)$, where U is as in Definition 9.

These five criteria are subjected to two different kinds of tests:

Error probability The error probability for a criterion is the probability that it will select a model that does not contain the distribution from which the data are sampled. We estimate the error probability through repeated sampling: in our experiments, samples are always drawn from a $\text{Poisson}[\mu]$ distribution with probability p , or from a $\text{Geom}[\mu]$ distribution with probability $1 - p$. Figure 2 shows the error probability as a function of the sample size on a log scale, for various values of p and μ . After the first two graphs, we plot the ratio of the error probability of a criterion with the error probability of the baseline “known mean” criterion: this allows for better distinction between the criteria.

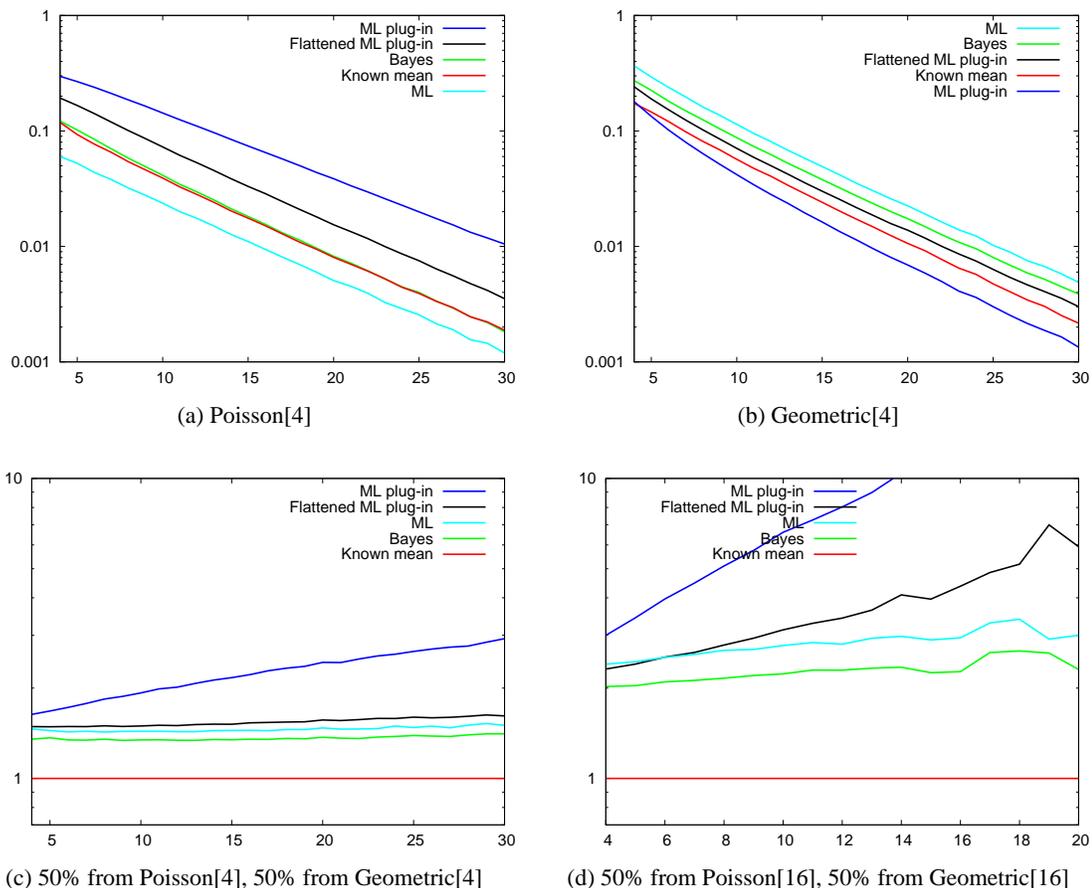


Figure 2: Error probability. For Figures (c) and (d), the error frequency is divided by the baseline, the error frequency of the Known mean criterion. Estimated using 10^6 trials.

Bias Let $\Delta_\mu(x^n)$ be the evidence in favor of the Poisson model according to the known mean criterion. For other criteria C , the quantity Δ_C can be interpreted as an *estimator* for Δ_μ . The bias of such an estimator is $E[\Delta_C(X^n) - \Delta_\mu(X^n)]$, where the expectation is taken under the true distribution. We subsequently estimate this bias for all criteria by calculating the average over many trials. The results are in Figure 3.

6.2 Discussion

In order to establish a context to discuss the behavior of the flattened ML plug-in criterion, we first briefly summarize the conclusions from (De Rooij & Grünwald, 2006), which still apply to the current experiments.

- ML and ML plug-in exhibited worst performance; the Bayesian criterion performed reasonably on all tests.
- We found that the ML criterion consistently displays the largest bias in favor of Poisson. Figure 3 shows how on average, for ML we obtained at least 0.4 nats more evidence in favor of the Poisson model than for known mean. The Poisson model appears to have a greater descriptive power, even though the two models have the same number of parameters: intuitively, the Poisson model allows more information about the data to be stored in the parameter estimate.
- In all graphs in Figure 2 one can observe the unusual slope of the error rate line of the ML plug-in criterion, which clearly favors the geometric distribution. This is very undesirable for model selection, because the error rate when data are sampled from Poisson with probability p and from geometric with probability $1 - p$, is dominated by the worst of the two cases, i.e. the case that the data are Poisson distributed. This explains why the error rate is so poor in the case where $p = 0.5$ (Figures 2c and 2d). The bias is visible more explicitly in Figure 3, where ML plug-in can be observed to become more and more favorable to the geometric model as the sample size increases, regardless of whether the data were sampled from a Poisson or geometric distribution.

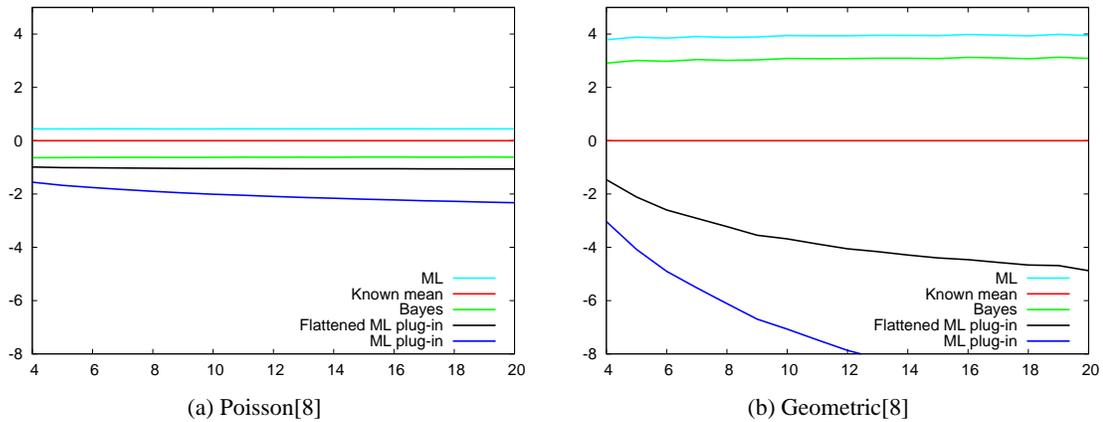


Figure 3: The classification bias in favor of the Poisson model in nats, estimated using 10^5 trials.

The new experiments also include results for the new flattened ML plug-in criterion. Figure 2 shows that, compared to ML plug-in, the slope of the error probability line for the flattened ML plug-in estimator is much closer to that of known mean. Nevertheless, when the mean is increased in subfigure (d), we see that the error probability seems to go down at a somewhat slower rate than it does for the Bayes and ML criteria.

In Figure 3 we find that, like ML plug-in, flattened ML plug-in is biased in favor of the geometric model. If the data are geometric, then this bias increases with sample size, as it does for ML plug-in, albeit at a slower rate. However, for Poisson data most of this effect appears to have been suppressed. This means that the probability that Poisson data are incorrectly judged to be geometric never becomes much larger than for other criteria, regardless of sample size. So for model selection purposes, the bias is acceptable.

In conclusion, the flattened ML plug-in criterion does indeed seem to provide a substantial improvement in model selection performance over the ML plug-in criterion. That said, the bias in favor of the geometric model has not completely vanished, which may be because of the $O(1)$ terms in the redundancy of the estimator which we did not analyze. The Bayesian criterion is clearly somewhat more reliable, but may be too computationally intensive depending on the considered models.

7 Conclusion

Given a model (set of probability distributions) \mathcal{M} , the maximum likelihood estimator $\hat{\theta}(x^n)$ based on past observations $x^n = x_1, \dots, x_n$ indexes a distribution that is a natural and easy to compute candidate for prediction of the next observation. However, previous work shows that if the data generating distribution P^* is not in the model, then such a “ML plug-in” prediction strategy yields suboptimal expected regret: unlike for other prediction strategies, such as Bayesian prediction, the expected regret is *not* $(k/2) \log n + O(1)$, where k is the number of parameters in the model. This is a serious problem when the “ML plug-in” strategy is used for model selection: there, by its very nature, the possibility that $P^* \notin \mathcal{M}$ deserves serious consideration.

To address this issue, we described a simple “flattening” of the ML distribution and related predictors, using which the optimal worst case *individual sequence* regret of $(k/2) \log n + O(1)$ can be achieved, for exponential family models and bounded outcome spaces (Theorem 11 on page 6). For unbounded spaces, we provided an almost-sure result (Theorem 14 on page 9). In Section 6, we subjected the new prediction strategy to the same model selection experiments that showed the ML plug-in strategy to be suboptimal, obtaining a major improvement in performance.

References

- Akaike, H. (1974). A new look at statistical model identification. *IEEE Trans. Autom. Contr.*, 19, 716–723.
- Azoury, K., & Warmuth, M. (2001). Relative loss bounds for on-line density estimation with the exponential family of distributions. *J. of Machine Learning*, 43, 211–246. Special issue on Theoretical Advances in On-Line Learning, Game Theory and Boosting, edited by Y. Singer.
- Barndorff-Nielsen, O. (1978). *Information and exponential families in statistical theory*. Chichester, UK: Wiley.
- Barron, A., Rissanen, J., & Yu, B. (1998). The minimum description length principle in coding and modeling. *IEEE Trans. Inf. Th.*, 44, 2743–2760. Special Commemorative Issue: Inf. Theory: 1948-1998.

- Cesa-Bianchi, N., & Lugosi, G. (2001). Worst-case bounds for the logarithmic loss of predictors. *J. of Machine Learning*, 43, 247–264.
- Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning and games*. Cambridge University Press.
- Corcuera, J., & Giummolè, F. (1999). A generalized Bayes rule for prediction. *Scandinavian J. of Statistics*, 26, 265–279.
- Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. John Wiley.
- Dawid, A. (1984). Present position and potential developments: Some personal views, statistical theory, the prequential approach. *J. Royal Stat.Soc., Ser. A*, 147, 278–292.
- De Rooij, S., & Grünwald, P. D. (2005). MDL model selection using the ML plug-in code. *Proc. 2005 International Symposium on Inf. Th. (ISIT 2005)*. Adelaide, Australia.
- De Rooij, S., & Grünwald, P. D. (2006). An empirical study of MDL model selection with infinite parametric complexity. *J. of Mathematical Psychology*, 50, 180–192.
- Freund, Y. (1996). Predicting a binary sequence almost as well as the optimal biased coin. *Proc. 9th Conf. on Computational Learning Theory (COLT 1996)* (pp. 89–98).
- Grünwald, P. (2007). *The minimum description length principle*. Cambridge, MA: MIT Press.
- Grünwald, P., & Kotłowski, W. (2010). Prequential plug-in codes that achieve optimal redundancy rates even if the model is wrong. arXiv:1002.0757.
- Grünwald, P. D. (2005). MDL tutorial. *Advances in Minimum Description Length: Theory and Applications*. MIT Press.
- Grünwald, P. D., & de Rooij, S. (2005). Asymptotic log-loss of prequential maximum likelihood codes. *Proc. 18th Conf. on Computational Learning Theory (COLT 2005)* (pp. 652–667).
- Hemerly, E., & Davis, M. (1989). Strong consistency of the PLS criterion for order determination of autoregressive processes. *The Annals of Statistics*, 17, 941–946.
- Hutter, M., & Poland, J. (2005). Adaptive online prediction by following the perturbed leader. *J. of Machine Learning Research*, 6, 639–660.
- Kalai, A., & Vempala, S. (2003). Efficient algorithms for online decision. *Proc. 16th Conf. on Computational Learning Theory (COLT 2003)* (pp. 506–521). Berlin: Springer.
- Krichevsky, R., & Trofimov, V. (1981). The performance of universal encoding. *IEEE Trans. Inf. Th., IT-27*, 199–207.
- Li, L., & Yu, B. (2000). Iterated logarithmic expansions of the pathwise code lengths for exponential families. *IEEE Trans. Inf. Th., 46*, 2683–2689.
- Rissanen, J. (1984). Universal coding, information, prediction and estimation. *IEEE Trans. Inf. Th., 30*, 629–636.
- Rissanen, J. (1986). A predictive least squares principle. *IMA J. of Math. Contr. and Inf.*, 3, 211–222.
- Rissanen, J. (1989). *Stochastic complexity in statistical inquiry*. World Scientific Publishing Company.
- Rissanen, J. (1996). Fisher information and stochastic complexity. *IEEE Trans. Inf. Th., IT-42*, 40–47.
- Rissanen, J., & Roos, T. (2007). Conditional NML universal models. *Proc. Inf. Th. and Applications Workshop (ITA-07)* (pp. 337–341). IEEE Press.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6.
- Shtarkov, Y. (1987). Universal sequential coding of single messages. *Problems of Inf. Trans.*, 23, 175–186.
- Takimoto, E., & Warmuth, M. (2000). The last-step minimax algorithm. *Proc. 11th Conf. on Algorithmic Learning Theory (ALT 2000)*.
- Vidoni, P. (2008). Improved predictive model selection. *J. of Stat. Planning and Inference*, 138, 3713–3721.
- Wei, C. (1990). On predictive least squares principles. *The Annals of Statistics*, 20, 1–42.

Sequence prediction in realizable and non-realizable cases

Daniil Ryabko
INRIA Lille-Nord Europe,
daniil@ryabko.net

Abstract

A sequence x_1, \dots, x_n, \dots of discrete-valued observations is generated according to some unknown probabilistic law (measure) μ . After observing each outcome, it is required to give the conditional probabilities of the next observation. The realizable case is when the measure μ belongs to an arbitrary but known class \mathcal{C} of process measures. The non-realizable case is when μ is completely arbitrary, but the prediction performance is measured with respect to a given set \mathcal{C} of process measures. We are interested in the relations between these problems and between their solutions, as well as in characterizing the cases when a solution exists, and finding these solutions. We show that if the quality of prediction is measured by total variation distance, then these problems coincide, while if it is measured by expected average KL divergence, then they are different. For some of the formalizations we also show that when a solution exists, it can be obtained as a Bayes mixture over a countable subset of \mathcal{C} . As an illustration to the general results obtained, we show that a solution to the non-realizable case of the sequence prediction problem exists for the set of all finite-memory processes, but does not exist for the set of all stationary processes.

1 Introduction

A sequence x_1, \dots, x_n, \dots of discrete-valued observations ($x_i \in \mathcal{X}$, \mathcal{X} is finite) is generated according to some unknown probabilistic law (measure). That is, μ is a probability measure on the space $\Omega = (\mathcal{X}^\infty, \mathcal{B})$ of one-way infinite sequences (here \mathcal{B} is the usual Borel σ -algebra). After each new outcome x_n is revealed, it is required to predict conditional *probabilities* of the next observation $x_{n+1} = a$, $a \in \mathcal{X}$, given the past x_1, \dots, x_n . Since a predictor ρ is required to give conditional probabilities $\rho(x_{n+1} = a | x_1, \dots, x_n)$ for all possible histories x_1, \dots, x_n , it defines itself a probability measure on the space Ω of one-way infinite sequences. In other words, a probability measure on Ω can be considered both as a data-generating mechanism and as a predictor.

Therefore, given a set \mathcal{C} of probability measures on Ω , one can ask two kinds of questions about it. First, does there exist a predictor ρ , whose forecast probabilities converge (in a certain sense) to the μ -conditional probabilities, if an arbitrary $\mu \in \mathcal{C}$ is chosen to generate the data? Here we assume that the “true” measure that generates the data belongs to the set \mathcal{C} of interest, and would like to construct a predictor that predicts all measures in \mathcal{C} . The second type of questions is as follows: does there exist a predictor that predicts at least as well as any predictor $\rho \in \mathcal{C}$, if the measure that generates the data comes possibly from outside of \mathcal{C} ? Therefore, here we consider elements of \mathcal{C} as predictors, and we would like to combine their predictive properties, if this is possible. Note that in this setting the two questions above concern the same object: a set \mathcal{C} of probability measures on Ω .

Each of these two questions, the realizable and non-realizable one, have enjoyed much attention in the literature; the setting for the non-realizable case is usually slightly different, which is probably why it has not (to the best of the author’s knowledge) been studied as another facet of the realizable case. The realizable case traces back to Laplace, who has considered the problem of predicting outcomes of a series of independent tosses of a biased coin. That is, he has considered the case when the set \mathcal{C} is that of all i.i.d. process measures. Other classical examples studied are the set of all computable (or semi-computable) measures [Solomonoff, 1978], the set of k -order Markov and finite-memory processes (e.g. [Krichevsky, 1993]) and the set of all stationary processes [Ryabko, 1988]. The general question of finding predictors for an arbitrary given set \mathcal{C} of process measures has been

addressed in [Ryabko and Hutter, 2007, Ryabko and Hutter, 2008, Ryabko, 2010]; the latter work shows that when a solution exists it can be obtained as a Bayes mixture over a countable subset of \mathcal{C} .

The non-realizable case is usually studied in a slightly different, non-probabilistic, setting. We refer to [Cesa-Bianchi and Lugosi, 2006] for a comprehensive overview. It is usually assumed that the observed sequence of outcomes is an arbitrary (deterministic) sequence; it is required not to give conditional probabilities, but just deterministic guesses (although these guesses can be selected using randomisation). Predictions result in a certain loss, which is required to be small as compared to the loss of a given set of reference predictors (experts) \mathcal{C} . The losses of the experts and the predictor are observed after each round. In this approach, it is mostly assumed that the set \mathcal{C} is finite or countable. The main difference with the formulation considered in this work is that we require a predictor to give probabilities, and thus the loss is with respect to something never observed (probabilities, not outcomes). The loss itself is not completely observable in our setting. In this sense our non-realizable version of the problem is more difficult. Assuming that the data generating mechanism is probabilistic, even if it is completely unknown, makes sense in such problems as, for example, game playing, or market analysis. In these cases one may wish to assign smaller loss to those models or experts who give probabilities closer to the correct ones (which are never observed), even though different probability forecasts can often result in the same action. Aiming at predicting probabilities of outcomes also allows us to abstract from the actual use of the predictions (e.g. making bets) and thus from considering losses in a general form; instead, we can concentrate on the form of losses (measuring the discrepancy between the forecast and true probabilities) which are more convenient for the analysis. In this latter respect, the problems we consider are easier than those considered in prediction with expert advice. (However, in principle nothing restricts us to considering the simple losses that we chose; they are just a convenient choice.) Noteworthy, the probabilistic approach also makes the machinery of probability theory applicable, hopefully making the problem easier.

In this work we consider two measures of the quality of prediction. The first one is the total variation distance, which measures the difference between the forecast and the “true” conditional probabilities of all future events (not just the probability of the next outcome). The second one is expected (over the data) average (over time) Kullback-Leibler divergence. Requiring that predicted and true probabilities converge in total variation is very strong; in particular, this is possible if [Blackwell and Dubins, 1962] and only if [Kalai and Lehrer, 1994] the process measure generating the data is absolutely continuous with respect to the predictor. The latter fact makes the sequence prediction problem relatively easy to analyse. Here we investigate what can be paralleled for the other measure of prediction quality (average KL divergence), which is much weaker, and thus allows for solutions for the cases of much larger sets \mathcal{C} of process measures (considered either as predictors or as data generating mechanisms).

Having introduced our measures of prediction quality, we can further break the non-realizable case into two problems. The first one is as follows. Given a set \mathcal{C} of predictors, we want to find a predictor whose prediction error converges to zero if there is at least one predictor in \mathcal{C} whose prediction error converges to zero; we call this problem simply the “non-realizable” case, or Problem 2 (leaving the name “Problem 1” to the realizable case). The second problem is the “fully agnostic” problem: it is to make the prediction error asymptotically as small as that of the best (for the given process measure generating the data) predictor in \mathcal{C} (we call this Problem 3). Thus, we now have three problems about a set of process measures \mathcal{C} to address.

We show that if the quality of prediction is measured in total variation, then all the three problems coincide: any solution to any one of them is a solution to the other two. For the case of expected average KL divergence, all the three problems are different: the realizable case is strictly easier than non-realizable (Problem 2), which is, in turn, strictly easier than the fully agnostic case (Problem 3). We then analyse which results concerning prediction in total variation can be transferred to which of the problems concerning prediction in average KL divergence. It was shown in [Ryabko, 2010] that, for the realizable case, if there is a solution for a given set of process measures \mathcal{C} , then a solution can also be obtained as a Bayesian mixture over a countable subset of \mathcal{C} ; this holds both for prediction in total variation and in expected average KL divergence. Here we show that this result also holds true for the (non-realizable) case of Problem 2, for prediction in expected average KL divergence. For the fully agnostic case of Problem 3, we show that separability with respect to a certain topology given by KL divergence is a sufficient (though not a necessary) condition for the existence of a predictor. This is used to demonstrate that there is a solution to this problem for the set of all finite-memory process measures, complementing similar results obtained earlier in different settings. On the other hand, we show that there is no solution to this problem for the set of all stationary process measures, in contrast to a result of [Ryabko, 1988] which gives a solution to the

realizable case of this problem (that is, a predictor whose expected average KL error goes to zero if any stationary process is chosen to generate the data).

2 Preliminaries

Let \mathcal{X} be a finite set. The notation $x_{1..n}$ is used for x_1, \dots, x_n . We consider stochastic processes (probability measures) on $\Omega := (\mathcal{X}^\infty, \mathcal{B})$ where \mathcal{B} is the sigma-field generated by the cylinder sets $[x_{1..n}]$, $x_i \in \mathcal{X}$, $n \in \mathbb{N}$ and $[x_{1..n}]$ is the set of all infinite sequences that start with $x_{1..n}$. For a finite set A denote $|A|$ its cardinality. We use \mathbf{E}_μ for expectation with respect to a measure μ .

Next we introduce the measures of the quality of prediction used in this paper. For two measures μ and ρ we are interested in how different the μ - and ρ -conditional probabilities are, given a data sample $x_{1..n}$. Introduce the (*conditional total variation distance*)

$$v(\mu, \rho, x_{1..n}) := \sup_{A \in \mathcal{B}} |\mu(A|x_{1..n}) - \rho(A|x_{1..n})|,$$

if $\mu(x_{1..n}) \neq 0$ and $\rho(x_{1..n}) \neq 0$, and $v(\mu, \rho, x_{1..n}) = 1$ otherwise.

Definition 1 We say that ρ predicts μ in total variation if

$$v(\mu, \rho, x_{1..n}) \rightarrow 0 \text{ } \mu\text{-a.s.}$$

This convergence is rather strong. In particular, it means that ρ -conditional probabilities of arbitrary far-off events converge to μ -conditional probabilities. Moreover, ρ predicts μ in total variation if [Blackwell and Dubins, 1962] and only if [Kalai and Lehrer, 1994] μ is absolutely continuous with respect to ρ . Denote \geq_{tv} the relation of absolute continuity (that is, $\rho \geq_{tv} \mu$ if μ is absolutely continuous with respect to ρ).

Thus, for a class \mathcal{C} of measures there is a predictor ρ that predicts every $\mu \in \mathcal{C}$ in total variation if and only if every $\mu \in \mathcal{C}$ has a density with respect to ρ . Although such sets of processes are rather large, they do not include even such basic examples as the set of all Bernoulli i.i.d. processes. That is, there is no ρ that would predict in total variation every Bernoulli i.i.d. process measure δ_p , $p \in [0, 1]$, where p is the probability of 0. Therefore, perhaps for many (if not most) practical applications this measure of the quality of prediction is too strong, and one is interested in weaker measures of performance.

For two measures μ and ρ introduce the *expected cumulative Kullback-Leibler divergence (KL divergence)* as

$$d_n(\mu, \rho) := \mathbf{E}_\mu \sum_{t=1}^n \sum_{a \in \mathcal{X}} \mu(x_t = a|x_{1..t-1}) \log \frac{\mu(x_t = a|x_{1..t-1})}{\rho(x_t = a|x_{1..t-1})}, \quad (1)$$

In words, we take the expected (over data) cumulative (over time) KL divergence between μ - and ρ -conditional (on the past data) probability distributions of the next outcome.

Definition 2 We say that ρ predicts μ in expected average KL divergence if

$$\frac{1}{n} d_n(\mu, \rho) \rightarrow 0.$$

This measure of performance is much weaker, in the sense that it requires good predictions only one step ahead, and not on every step but only on average; also the convergence is not with probability 1 but in expectation. With prediction quality so measured, predictors exist for relatively large classes of measures; most notably, [Ryabko, 1988] provides a predictor which predicts every stationary process in expected average KL divergence. We will use the following well-known identity

$$d_n(\mu, \rho) = - \sum_{x_{1..n} \in \mathcal{X}^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})}, \quad (2)$$

where on the right-hand side we have simply the KL divergence between measures μ and ρ restricted to the first n observations.

Thus, the results of this work will be established with respect to two very different measures of prediction quality, one of which is very strong and the other rather weak. This suggests that the facts established reflect some fundamental properties of the problem of prediction, rather than those pertinent to particular measures of performance. On the other hand, it remains open to extend the results below to different measures of performance.

Definition 3 Introduce the following classes of process measures: \mathcal{P} the set of all process measures, \mathcal{D} the set of all degenerate discrete process measures, \mathcal{S} the set of all stationary processes, and \mathcal{M}_k the set of all stationary measures with memory not greater than k (k -order Markov processes, with \mathcal{M}_0 being the set of all i.i.d. processes):

$$\mathcal{D} := \{\mu \in \mathcal{P} : \exists x \in \mathcal{X}^\infty \mu(x) = 1\}, \quad (3)$$

$$\mathcal{S} := \{\mu \in \mathcal{P} : \forall n, k \geq 0 \forall a_{1..n} \in \mathcal{X}^n \mu(x_{1..n} = a_{1..n}) = \mu(x_{1+k..n+k} = a_{1..n})\}. \quad (4)$$

$$\mathcal{M}_k := \left\{ \mu \in \mathcal{S} : \forall n \geq 0 \forall a \in \mathcal{X} \forall a_{1..n} \in \mathcal{X}^n \right. \\ \left. \mu(x_{n+1} = a | x_{1..n} = a_{1..n}) = \mu(x_{k+1} = a | x_{1..k} = a_{1..k}) \right\}, \quad (5)$$

Abusing the notation, we will sometimes use elements of \mathcal{D} and \mathcal{X}^∞ interchangeably. The following simple statement (whose proof is obvious) will be used repeatedly in the examples.

Lemma 4 For every $\rho \in \mathcal{P}$ there exists $\mu \in \mathcal{D}$ such that $d_n(\mu, \rho) \geq n \log |\mathcal{X}|$ for all $n \in \mathbb{N}$.

3 Sequence prediction problems

For the two notions of predictive quality introduced, we can now start stating formally the sequence prediction problems.

Problem 1 (realizable case). Given a set of probability measures \mathcal{C} , find a measure ρ such that ρ predicts in total variation (expected average KL divergence) every $\mu \in \mathcal{C}$, if such a ρ exists.

Thus, Problem 1 is about finding a predictor for the case when the process generating the data is known to belong to a given class \mathcal{C} . The set \mathcal{C} here is a set of measures generating the data. Next let us formulate the questions about \mathcal{C} as a set of predictors.

Problem 2 (non-realizable case). Given a set of process measures (predictors) \mathcal{C} , find a process measure ρ such that ρ predicts in total variation (in expected average KL divergence) every measure $\nu \in \mathcal{P}$ such that there is $\mu \in \mathcal{C}$ which predicts (in the same sense) ν .

While Problem 2 is already quite general, it does not yet address what can be called the fully agnostic case: if nothing at all is known about the process ν generating the data, it means that there may be no $\mu \in \mathcal{C}$ such that μ predicts ν , and then, even if we have a solution ρ to the Problem 2, we still do not know what the performance of ρ on ν is going to be, compared to the performance of the predictors from \mathcal{C} . To address the fully agnostic case, we have to introduce the notion of loss.

Definition 5 Introduce the almost sure total variation loss of ρ with respect to μ

$$l_{tv}(\mu, \rho) := \inf\{\alpha \in [0, 1] : \limsup_{n \rightarrow \infty} v(\mu, \rho, x_{1..n}) \leq \alpha \text{ } \mu\text{-a.s.}\},$$

and the asymptotic KL loss

$$l_{KL}(\nu, \rho) := \limsup_{n \rightarrow \infty} \frac{1}{n} d_n(\nu, \rho).$$

We can now formulate the fully agnostic version of the sequence prediction problem.

Problem 3. Given a set of process measures (predictors) \mathcal{C} , find a process measure ρ such that ρ predicts at least as well as any μ in \mathcal{C} , if any process measure $\nu \in \mathcal{P}$ is chosen to generate the data: $l(\nu, \rho) \leq l(\nu, \mu)$ for every $\nu \in \mathcal{P}$ and every $\mu \in \mathcal{C}$, where $l(\cdot, \cdot)$ is either $l_{tv}(\cdot, \cdot)$ or $l_{KL}(\cdot, \cdot)$.

The three problems just formulated represent different conceptual approaches to the sequence prediction problem. Let us illustrate the difference by the following informal example. Suppose that the set \mathcal{C} is that of all (ergodic, finite-state) Markov chains. Markov chains being a familiar object in probability and statistics, we can easily construct a predictor ρ that predicts every $\mu \in \mathcal{C}$ (for example, in expected average KL divergence, see [Krichevsky, 1993]). That is, if we know that the process μ generating the data is Markovian, we know that our predictor is going to perform well. This is the realizable case of Problem 1. In reality, rarely can we be sure that the Markov assumption holds true for the data at hand. We may believe, however, that it is still a reasonable assumption, in the sense that there is a Markovian model which, for our purposes (for the purposes of prediction), is a good model of the data. Thus we may assume that there is a Markov model (a predictor) that predicts well the process that we observe, and we would like to combine the predictive qualities of all these Markov models. This is the “non-realizable” case of Problem 2. Note that this problem is more difficult than the first one; in particular, a process ν generating the data may be singular with respect to any Markov process, and still be well predicted (in the sense of expected average

KL divergence, for example) by some of them. Still, here we are making some assumptions about the process generating the data, and if these assumptions are wrong, then we do not know anything about the performance of our predictor. Thus we may ultimately wish to acknowledge that we do not know anything at all about the data; we still know a lot about Markov processes, and we would like to use this knowledge on our data. If there is anything at all Markovian in it (that is, anything that can be captured by a Markov model), then we would like our predictor to use it. In other words, we want to have a predictor that predicts any process measure whatsoever (at least) as well as any Markov predictor. This is the “fully agnostic” case of Problem 3.

Of course, Markov processes were just mentioned as an example, while in this work we are only concerned with the most general case of arbitrary unknown (uncountable) sets \mathcal{C} of process measures.

The following statement is rather obvious.

Proposition 1 *Any solution to Problem 3 is a solution to Problem 2, and any solution to Problem 2 is a solution to Problem 1.*

Despite the conceptual differences in formulations, it may be somewhat unclear whether the three problems are indeed different. It appears that this depends on the measure of predictive quality chosen: for the case of prediction in total variation distance, all the three problems coincide, while for the case of prediction in expected average KL divergence, they are different.

4 Prediction in total variation

As it was mentioned, if a measure μ is absolutely continuous with respect to a measure ρ if and only if ρ predicts μ in total variation distance, [Blackwell and Dubins, 1962, Kalai and Lehrer, 1994]. This reduces the study of at least Problem 1 for total variation distance to studying the relation of absolute continuity. Introduce the notation $\rho \geq_{KL} \mu$ for this relation.

Let us briefly recall some facts we know about \geq_{tv} ; details can be found, for example, in [Plesner and Rokhlin, 1946]. Let $[\mathcal{P}]_{tv}$ denote the set of equivalence classes of \mathcal{P} with respect to \geq_{tv} , and for $\mu \in [\mathcal{P}]_{tv}$ denote $[\mu]$ the equivalence class that contains μ . Two elements $\sigma_1, \sigma_2 \in [\mathcal{P}]_{tv}$ (or $\sigma_1, \sigma_2 \in \mathcal{P}$) are called disjoint (or singular) if there is no $\nu \in [\mathcal{P}]_{tv}$ such that $\sigma_1 \geq_{tv} \nu$ and $\sigma_2 \geq_{tv} \nu$; in this case we write $\sigma_1 \perp_{tv} \sigma_2$. We write $[\mu_1] + [\mu_2]$ for $[1/2(\mu_1 + \mu_2)]$. Every pair $\sigma_1, \sigma_2 \in [\mathcal{P}]_{tv}$ has a supremum $\sup(\sigma_1, \sigma_2) = \sigma_1 + \sigma_2$. Introducing into $[\mathcal{P}]_{tv}$ an extra element 0 such that $\sigma \geq_{tv} 0$ for all $\sigma \in [\mathcal{P}]_{tv}$, we can state that for every $\rho, \mu \in [\mathcal{P}]_{tv}$ there exists a unique pair of elements μ_s and μ_a such that $\mu = \mu_a + \mu_s$, $\rho \geq \mu_a$ and $\rho \perp_{tv} \mu_s$. (This is a form of Lebesgue decomposition.) Moreover, $\mu_a = \inf(\rho, \mu)$. Thus, every pair of elements has a supremum and an infimum. Moreover, every bounded set of disjoint elements of $[\mathcal{P}]_{tv}$ is at most countable.

Furthermore, introduce the (unconditional) total variation distance between process measures.

Definition 6 (unconditional total variation distance) *Introduce the (unconditional) total variation distance*

$$v(\mu, \rho) := \sup_{A \in \mathcal{B}} |\mu(A) - \rho(A)|.$$

Known characterizations of sets \mathcal{C} bounded with respect to \geq_{KL} can now be related to our prediction problems 1-3 as follows.

Theorem 7 *Let $\mathcal{C} \subset \mathcal{P}$. The following statements about the set \mathcal{C} are equivalent.*

- (i) *There exists a solution to Problem 1 in total variation.*
- (ii) *There exists a solution to Problem 2 in total variation.*
- (iii) *There exists a solution to Problem 3 in total variation.*
- (iv) *\mathcal{C} is upper-bounded with respect to \geq_{tv} .*
- (v) *There exists a sequence $\mu_k \in \mathcal{C}$, $k \in \mathbb{N}$ such that for some (equivalently, for every) sequence of weights $w_k \in (0, 1]$, $k \in \mathbb{N}$ such that $\sum_{k \in \mathbb{N}} w_k = 1$, the measure $\nu = \sum_{k \in \mathbb{N}} w_k \mu_k$ satisfies $\nu \geq_{tv} \mu$ for every $\mu \in \mathcal{C}$.*
- (vi) *\mathcal{C} is separable with respect to the total variation distance.*
- (vii) *Let $\mathcal{C}^+ := \{\mu \in \mathcal{P} : \exists \rho \in \mathcal{C} \rho \geq_{tv} \mu\}$. Every disjoint (with respect to \geq_{tv}) subset of \mathcal{C}^+ is at most countable.*

Moreover, every solution to any of the Problems 1-3 is a solution to the other two, as is any upper bound for \mathcal{C} . The sequence μ_k in the statement (v) can be taken to be any dense (in the total variation distance) countable subset of \mathcal{C} (cf. (vi)), or any maximal disjoint (with respect to \geq_{tv}) subset of \mathcal{C}^+ of statement (vii), in which every measure that is not in \mathcal{C} is replaced by any measure from \mathcal{C} that dominates it.

Proof: The implications $(i) \Leftarrow (ii) \Leftarrow (iii)$ are obvious (see Proposition 1). The implication $(i) \Rightarrow (iv)$ is a reformulation of the result of [Blackwell and Dubins, 1962]. The converse (and hence $(iv) \Rightarrow (i)$) was established in [Kalai and Lehrer, 1994]. $(i) \Rightarrow (ii)$ follows from the equivalence $(i) \Leftrightarrow (iv)$ and the transitivity of \geq_{tv} ; $(i) \Rightarrow (iii)$ follows from this equivalence and from Lemma 8 below. The equivalence of (v), (vi), and (i) was established in [Ryabko, 2010]. The equivalence of (iv) and (vii) was proven in [Plesner and Rokhlin, 1946]. The concluding statements of the theorem are easy to demonstrate from the results cited above. ■

The following lemma is an easy consequence of [Blackwell and Dubins, 1962].

Lemma 8 *Let μ, ρ be two process measures. Then $v(\mu, \rho, x_{1..n})$ converges to either 0 or 1 with μ -probability 1.*

Proof: Assume that μ is not absolutely continuous with respect to ρ (the other case is covered by [Blackwell and Dubins, 1962]). By Lebesgue decomposition theorem, the measure μ admits a representation $\mu = \alpha\mu_a + (1 - \alpha)\mu_s$ where $\alpha \in [0, 1]$ and the measures μ_a and μ_s are such that μ_a is absolutely continuous with respect to ρ and μ_s is singular with respect to ρ . Let W be such a set that $\mu_a(W) = \rho(W) = 1$ and $\mu_s(W) = 0$. Note that $\mu_a = \mu|_W$ and $\mu_s = \mu|_{\mathcal{X}^\infty \setminus W}$. From [Blackwell and Dubins, 1962] we have $v(\mu_a, \rho, x_{1..n}) \rightarrow 0$ μ_a -a.s., as well as $v(\mu_a, \mu, x_{1..n}) \rightarrow 0$ μ_a -a.s. and $v(\mu_s, \mu, x_{1..n}) \rightarrow 0$ μ_s -a.s. Moreover, $v(\mu_s, \rho, x_{1..n}) \geq |\mu_s(W|x_{1..n}) - \rho(W|x_{1..n})| = 1$ so that $v(\mu_s, \rho, x_{1..n}) \rightarrow 1$ μ_s -a.s. We have

$$v(\mu, \rho, x_{1..n}) \leq v(\mu, \mu_a, x_{1..n}) + v(\mu_a, \rho, x_{1..n}) = I$$

and

$$v(\mu, \rho, x_{1..n}) \geq -v(\mu, \mu_s, x_{1..n}) + v(\mu_s, \rho, x_{1..n}) = II$$

for $x_{1..n} \in W$ we have $I \rightarrow 0$ μ -a.s., and for $x_{1..n} \notin W$ we have $II \rightarrow 1$ μ -a.s., which concludes the proof. ■

Using Lemma 8 we could also define *expected* (rather than almost sure) total variation loss of ρ with respect to μ , as the probability that $v(\mu, \rho)$ converges to 1, and reformulate Problem 3 for this notion of loss. However, it is easy to see that for this reformulation Theorem 9 holds true as well.

Thus, we can see that for the case of prediction in total variation, all the sequence prediction problems formulated reduce to studying the relation of absolute continuity for process measures, and those families of measures that are absolutely continuous (have a density) with respect to some measure (a predictor). On the one hand, from statistical point of view such families are rather large: the assumption that the probabilistic law in question has a density with respect to some (nice) measure is a standard one in statistics. It should also be mentioned that such families can easily be uncountable. On the other hand, even such basic examples as the set of all Bernoulli i.i.d. measures does not allow for a predictor that predicts every measure in total variation. Indeed, all these processes are singular with respect to one another; in particular, each of the non-overlapping sets T_p of all sequences which have limiting fraction p of 0s has probability 1 with respect to one of the measures and 0 with respect to all others; since there are uncountably many of these measures, there is no measure ρ with respect to which they all would have a density (since such a measure should have $\rho(T_p) > 0$ for all p).

That is why we have to consider weaker notions of predictions; from these, prediction in expected average KL divergence is perhaps one of the weakest. The goal of the next sections is to see which of the properties that we have for total variation can be transferred (and in which sense) to the case of expected average KL divergence.

5 Prediction in expected average KL divergence

First of all we have to observe that for prediction in KL divergence Problems 1, 2, and 3 are different, as the following theorem shows. While the examples provided in the proof are artificial, there is a very important example illustrating the difference between Problem 1 and Problem 3 for expected average KL divergence: the set \mathcal{S} of all stationary processes, given in Theorem 15 in the end of this section.

Theorem 9 For the case of prediction in expected average KL divergence, Problems 1, 2 and 3 are different: there exists a set $\mathcal{C}_1 \subset \mathcal{P}$ for which there is a solution to Problem 1 but there is no solution to Problem 2, and there is a set $\mathcal{C}_2 \subset \mathcal{P}$ for which there is a solution to Problem 2 but there is no solution to Problem 3.

Proof: We have to provide two examples. Fix the binary alphabet $\mathcal{X} = \{0, 1\}$. For each deterministic sequence $t = t_1, t_2, \dots \in \mathcal{D}$ construct the process measure γ_t as follows: $\gamma_t(x_n = t_n | t_{1..n-1}) := 1 - \frac{1}{n}$ and for $x_{1..n-1} \neq t_{1..n-1}$ let $\gamma_t(x_n = 0 | x_{1..n-1}) = 1/2$, for all $n \in \mathbb{N}$. That is, γ_t is Bernoulli i.i.d. $1/2$ process measure strongly biased towards one deterministic sequence, t . Let also $\gamma(x_{1..n}) = 2^{-n}$ for all $x_{1..n} \in \mathcal{X}^n$, $n \in \mathbb{N}$ (the Bernoulli i.i.d. $1/2$). For the set $\mathcal{C}_1 := \{\gamma_t : t \in \mathcal{X}^\infty\}$ we have a solution to Problem 1: indeed, $d_n(\gamma_t, \gamma) \leq 1 = o(n)$. However, there is no solution to Problem 2. Indeed, for each $t \in \mathcal{D}$ we have $d_n(t, \gamma_t) = \log n = o(n)$ (that is, for every discrete measure there is an element of \mathcal{C}_1 which predicts it), while by Lemma 4 for every $\rho \in \mathcal{P}$ there exists $t \in \mathcal{D}$ such that $d_n(t, \rho) \geq n$ for all $n \in \mathbb{N}$ (that is, there is no predictor which predicts every measure that is predicted by at least one element of \mathcal{C}_1).

The second example is similar. For each deterministic sequence $t = t_1, t_2, \dots \in \mathcal{D}$ construct the process measure γ'_t as follows: $\gamma'_t(x_n = t_n | t_{1..n-1}) := 2/3$ and for $x_{1..n-1} \neq t_{1..n-1}$ let $\gamma'_t(x_n = 0 | x_{1..n-1}) = 1/2$, for all $n \in \mathbb{N}$. It is easy to see that γ is a solution to Problem 2 for the set $\mathcal{C}_2 := \{\gamma'_t : t \in \mathcal{X}^\infty\}$. However, there is no solution to Problem 3 for \mathcal{C}_2 . Indeed, for every $t \in \mathcal{D}$ we have $d_n(t, \gamma'_t) = n \log 3/2 + o(n)$. Therefore, if ρ is a solution to Problem 3 then $\limsup \frac{1}{n} d_n(t, \rho) \leq \log 3/2 < 1$ which contradicts Lemma 4. \blacksquare

Thus, prediction in expected average KL divergence turns out to be a more complicated matter than prediction in total variation. The next idea is to try and see which of the facts about prediction in total variation can be generalized to some of the problems concerning prediction in expected average KL divergence.

First, observe that for the case of prediction in total variation, the equivalence of Problems 1 and 2 was derived from the transitivity of the relation \geq_{tv} of absolute continuity. For the case of expected average KL divergence, the relation “ ρ predicts μ in expected average KL divergence” is not transitive (and Problems 1 and 2 are not equivalent). However, for Problem 2 we are interested in the following relation: ρ “dominates” μ if ρ predicts every ν such that μ predicts ν . This relation is transitive. Denote it by \geq_{KL}^0 .

Definition 10 (\geq_{KL}^0) We write $\rho \geq_{KL}^0 \mu$ if for every $\nu \in \mathcal{P}$ the equality $\limsup \frac{1}{n} d_n(\nu, \mu) = 0$ implies $\limsup \frac{1}{n} d_n(\rho, \mu) = 0$.

Similarly to \geq_{tv} , we can see that for any μ, ρ any strictly convex combination $\alpha\mu + (1 - \alpha)\rho$ is a supremum of $\{\rho, \mu\}$ with respect to \geq_{KL}^0 . Next we will obtain a characterization of predictability with respect to \geq_{KL}^0 similar to one of those obtained for \geq_{tv} .

The key observation is the following. If there is a solution to Problem 2 for a set \mathcal{C} , then a solution can be obtained as a Bayesian mixture over a countable subset of \mathcal{C} . For total variation, this is (v) of Theorem 7.

Theorem 11 Let \mathcal{C} be a set of probability measures on Ω . If there is a measure ρ such that $\rho \geq_{KL}^0 \mu$ for every $\mu \in \mathcal{C}$ (ρ is a solution to Problem 2), then there is a sequence $\mu_k \in \mathcal{C}$, $k \in \mathbb{N}$ such that $\sum_{k \in \mathbb{N}} w_k \mu_k \geq_{KL}^0 \mu$ for every $\mu \in \mathcal{C}$, where w_k are some positive weights.

The proof is deferred to Appendix. An analogous result for Problem 1 was established in [Ryabko, 2009]. (The proof of Theorem 11 is based on similar ideas, but is more involved.)

For the case of Problem 3, it remains open to prove a result similar to Theorem 11 (or statement (v) of Theorem 7). However, we can take a different route and extend another part of Theorem 7 to obtain a characterization of sets \mathcal{C} for which a solution to Problem 3 exists.

We have seen that in the case of prediction in total variation, separability with respect to the topology of this distance is a necessary and sufficient condition for the existence of a solution to Problems 1-3. In the case of expected average KL divergence the situation is somewhat different, since, first of all, (asymptotic average) KL divergence is not a metric. While one can introduce a topology based on it, separability with respect to this topology turns out to be a sufficient but not a necessary condition for the existence of a predictor, as is shown in the next theorem.

Definition 12 Define the distance $d_\infty(\mu_1, \mu_2)$ on process measures as follows

$$d_\infty(\mu_1, \mu_2) = \limsup_{n \rightarrow \infty} \sup_{x_{1..n} \in \mathcal{X}^n} \frac{1}{n} \left| \log \frac{\mu_1(x_{1..n})}{\mu_2(x_{1..n})} \right|. \quad (6)$$

Clearly, d_∞ is symmetric and transitive, but is not exact. Moreover, for every μ_1, μ_2 we have

$$\limsup_{n \rightarrow \infty} \frac{1}{n} d_n(\mu_1, \mu_2) \leq d_\infty(\mu_1, \mu_2). \quad (7)$$

The distance $d_\infty(\mu_1, \mu_2)$ measures the difference in behaviour of μ_1 and μ_2 on all individual sequences. Thus, using this distance to analyze Problem 3 is most close to the traditional approach to the non-realizable case, which is formulated in terms of predicting individual deterministic sequences.

Theorem 13 (i) *Let \mathcal{C} be a set of process measures. If \mathcal{C} is separable with respect to d_∞ then there is a solution to Problem 3 for \mathcal{C} , for the case of prediction in expected average KL divergence.*

(ii) *There exists a set of process measures \mathcal{C} such that \mathcal{C} is not separable with respect to d_∞ , but there is a solution to Problem 3 for this set, for the case of prediction in expected average KL divergence.*

Proof: For the first statement, let \mathcal{C} be separable and let $(\mu_k)_{k \in \mathbb{N}}$ be a dense countable subset of \mathcal{C} . Define $\nu := \sum_{k \in \mathbb{N}} w_k \mu_k$, where w_k are any positive summable weights. Fix any measure τ and any $\mu \in \mathcal{C}$. We will show that $\limsup_{n \rightarrow \infty} \frac{1}{n} d_n(\tau, \nu) \leq \limsup_{n \rightarrow \infty} \frac{1}{n} d_n(\tau, \mu)$. For every ε , find such a $k \in \mathbb{N}$ that $d_\infty(\mu, \mu_k) \leq \varepsilon$. We have

$$\begin{aligned} d_n(\tau, \nu) &\leq d_n(\tau, w_k \mu_k) = \mathbf{E}_\tau \log \frac{\tau(x_{1..n})}{\mu_k(x_{1..n})} - \log w_k \\ &= \mathbf{E}_\tau \log \frac{\tau(x_{1..n})}{\mu(x_{1..n})} + \mathbf{E}_\tau \log \frac{\mu(x_{1..n})}{\mu_k(x_{1..n})} - \log w_k \\ &\leq d_n(\tau, \mu) + \sup_{x_{1..n} \in \mathcal{X}^n} \log \left| \frac{\mu(x_{1..n})}{\mu_k(x_{1..n})} \right| - \log w_k. \end{aligned}$$

From this, dividing by n taking $\limsup_{n \rightarrow \infty}$ on both sides, we conclude

$$\limsup_{n \rightarrow \infty} \frac{1}{n} d_n(\tau, \nu) \leq \limsup_{n \rightarrow \infty} \frac{1}{n} d_n(\tau, \mu) + \varepsilon.$$

Since this holds for every $\varepsilon > 0$ the first statement is proven.

The second statement is proven by the following example. Let \mathcal{C} be the set of all deterministic sequences (measures concentrated on just one sequence) such that the number of 0s in the first n symbols is less than \sqrt{n} . Clearly, this set is uncountable. It is easy to check that $\mu_1 \neq \mu_2$ implies $d_\infty(\mu_1, \mu_2) = \infty$ for every $\mu_1, \mu_2 \in \mathcal{C}$, but the predictor ν given by $\nu(x_n = 0) = 1/n$ independently for different n , predicts every $\mu \in \mathcal{C}$ in expected average KL divergence. Since all elements of \mathcal{C} are deterministic, ν is also a solution to Problem 3 for \mathcal{C} . ■

Although simple, Theorem 13 can be used to establish the existence of a solution to Problem 3 for an important class of process measures: that of all processes with finite memory, as the next theorem shows. Results, similar to Theorem 14 are known in different settings, e.g. [Ziv and Lempel, 1978, Ryabko, 1984, Cesa-Bianchi and Lugosi, 1999] and others.

Theorem 14 *There exists a solution to Problem 3 for prediction in expected average KL divergence for the set of all finite-memory process measures $\mathcal{M} := \cup_{k \in \mathbb{N}} \mathcal{M}_k$.*

Proof: We will show that the set \mathcal{M} is separable with respect to d_∞ . Then the statement will follow from Theorem 13. It is enough to show that each set \mathcal{M}_k is separable with respect to d_∞ .

Observe that the family \mathcal{M}_k of k -order stationary binary-valued Markov processes is parametrized by $|\mathcal{X}|^{k+1} [0, 1]$ -valued parameters: probability of observing 0 after observing $x_{1..k}$, for each $x_{1..k} \in \mathcal{X}^k$. For each $k \in \mathbb{N}$ let μ_q^k , $q \in Q^{2^k}$ be the (countable) family of all stationary k -order Markov processes with rational values of all the parameters. We will show that this family is dense in \mathcal{M}_k . Indeed, for any $\mu_1, \mu_2 \in \mathcal{M}_k$ and every $x_{1..n} \in \mathcal{X}^n$ such that $\mu_i(x_{1..n}) \neq 0$, $i = 1, 2$, it is easy to see that

$$\frac{1}{n} \left| \log \frac{\mu_1(x_{1..n})}{\mu_2(x_{1..n})} \right| \leq 2 \log(a + \tau) \quad (8)$$

where $a = \inf_{x_{1..k}: \mu_i(x_{1..k}) \neq 0, i=1,2} \mu_i(x_{1..k})$ and $\tau := \inf_{x \in \mathcal{X}, x_{1..k} \in \mathcal{X}^k} |\mu_1(x|x_{1..k}) - \mu_2(x|x_{1..k})|$. Since the set μ_q^k , $q \in Q^{2^k}$ is dense in \mathcal{M}_k with respect to this parametrization, for each $\mu \in \mathcal{M}_k$ the expression (8) can be made arbitrary small for appropriate μ_q^k , so that \mathcal{M}_k is separable with respect to d_∞ . ■

Another important example is the set of all stationary process measures \mathcal{S} . This example also illustrates the difference between the prediction problems that we consider. For this set a solution to Problem 1 was given in [Ryabko, 1988]. In contrast, here we show that there is no solution to Problem 3 for \mathcal{S} .

Theorem 15 *There is no solution to Problem 3 for the set of all stationary processes \mathcal{S} .*

Proof: This proof is based on the construction similar to the one used in [Ryabko, 1988] to demonstrate impossibility of consistent prediction of stationary processes without Cesaro averaging.

Let m be a Markov chain with states $0, 1, 2, \dots$ and state transitions defined as follows. From each state $k \in \mathbb{N} \cup \{0\}$ the chain passes to the state $k + 1$ with probability $2/3$ and to the state 0 with probability $1/3$. It is easy to see that this chain possesses a unique stationary distribution on the set of states (see e.g. [Shiryaev, 1996]); taken as the initial distribution it defines a stationary ergodic process with values in $\mathbb{N} \cup \{0, 1\}$. Fix the ternary alphabet $\mathcal{X} = \{a, 0, 1\}$. For each sequence $t = t_1, t_2, \dots \in \{0, 1\}^\infty$ define the process μ_t as follows. It is a deterministic function of the chain m . If the chain is in the state 0 then the process μ_t outputs a ; if the chain m is in the state $k > 0$ then the process outputs t_k . That is, we have defined a hidden Markov process which in the state 0 of the underlying Markov chain always outputs a , while in other states it outputs either 0 or 1 according to the the sequence t .

To show that there is no solution to Problem 3 for \mathcal{S} , we will show that there is no solution to Problem 3 for the smaller set $\mathcal{C} := \{\mu_t : t \in \{0, 1\}^\infty\}$. Indeed, for any $t \in \{0, 1\}^\infty$ we have $d_n(t, \mu_t) = n \log 3/2 + o(n)$. Then if ρ is a solution to Problem 3 for \mathcal{C} we should have $\limsup_{n \rightarrow \infty} \frac{1}{n} d_n(t, \rho) \leq \log 3/2 < 1$ for every $t \in \mathcal{D}$, which contradicts Lemma 4. ■

From the proof Theorem 15 one can see that, in fact, the statement that is proven is stronger: there is no solution to Problem 3 for the set of all functions of stationary ergodic countable-state Markov chains. We conjecture that a solution to Problem 2 exists for the latter set, but not for the set of all stationary processes.

6 Discussion

It has been long realized that the so-called probabilistic and agnostic (adversarial, non-stochastic, deterministic) settings of the problem of sequential prediction are strongly related. This has been most evident from looking at the solutions to these problems, which are usually based on the same ideas. Here we have proposed a formulation of the agnostic problem as a non-realizable case of the probabilistic problem. While being very close to the traditional one, this setting allows us to directly compare the two problems. As a somewhat surprising result, we can see that whether the two problems are different depends on the measure of performance chosen: in the case of prediction in total variation distance they coincide, while in the case of prediction in expected average KL divergence they are different. In the latter case, the distinction becomes particularly apparent on the example of stationary processes: while a solution to the realizable problem has long been known, here we have shown that there is no solution to the agnostic version of this problem. The new formalization also allowed us to introduce another problem that lies in between the realizable and the fully agnostic problems: given a class of process measures \mathcal{C} , find a predictor that is predicts asymptotically optimal every measure for which at least one of the measures in \mathcal{C} is asymptotically optimal (Problem 2). This problem is less restrictive than the fully agnostic one (in particular, it is not concerned with the behaviour of a predictor on every deterministic sequence) but at the same time the solutions to this problem have performance guarantees far outside the model class considered.

Since the problem formulations presented here are mostly new (at least, in such a general form), it is not surprising that there are many questions left open. A promising route to obtain new results seems to be to first analyse the case of prediction in total variation, which amounts to studying the relation of absolute continuity and singularity of probability measures, and then to try and find analogues in less restrictive (and thus more interesting and difficult) cases of predicting only the next observation, possibly with Cesaro averaging. This is the approach that we took in this work. Here it is interesting to find properties common to all or most of the prediction problems (in total variation as well as with respect to other measures of the performance). A candidate is the “countable Bayes” property of Theorem 11: if there is a solution to a given sequence prediction problem for a set \mathcal{C} , then a solution can be obtained as a mixture over a suitable countable subset of \mathcal{C} .

Another direction for future research concerns finite-time performance analysis. In this work we have adopted the asymptotic approach to the prediction problem, ignoring the behaviour of predictors before asymptotic. While for prediction in total variation it is a natural choice, for other

measures of performance, including average KL divergence, it is clear that Problems 1-3 admit non-asymptotic formulations. It is also interesting what are the relations between performance guarantees that can be obtained in non-asymptotic formulations of Problems 1-3.

Appendix: Proof of Theorem 11

Proof: Define the weights $w_k := wk^{-2}$, where w is the normalizer $6/\pi^2$. Define the sets C_μ as the set of all measures $\tau \in \mathcal{P}$ such that μ predicts τ in expected average KL divergence. Let $\mathcal{C}^+ := \cup_{\mu \in \mathcal{C}} C_\mu$. For each $\tau \in \mathcal{C}^+$ let $p(\tau)$ be any (fixed) $\mu \in \mathcal{C}$ such that $\tau \in C_\mu$. In other words, \mathcal{C}^+ is the set of all measures that are predicted by some of the measures in \mathcal{C} , and for each measure τ in \mathcal{C}^+ we designate one ‘‘parent’’ measure $p(\tau)$ from \mathcal{C} such that $p(\tau)$ predicts τ .

Step 1. For each $\mu \in \mathcal{C}^+$ let δ_n be any monotonically increasing function such that $\delta_n(\mu) = o(n)$ and $d_n(\mu, p(\mu)) = o(\delta_n(\mu))$. Define the sets

$$U_\mu^n := \left\{ x_{1..n} \in \mathcal{X}^n : \mu(x_{1..n}) \geq \frac{1}{n} \rho(x_{1..n}) \right\}, \quad (9)$$

$$V_\mu^n := \left\{ x_{1..n} \in \mathcal{X}^n : p(\mu)(x_{1..n}) \geq 2^{-\delta_n(\mu)} \mu(x_{1..n}) \right\}, \quad (10)$$

and

$$T_\mu^n := U_\mu^n \cap V_\mu^n. \quad (11)$$

We will upper-bound $\mu(T_\mu^n)$. First, using Markov’s inequality, we derive

$$\mu(\mathcal{X}^n \setminus U_\mu^n) = \mu \left(\frac{\rho(x_{1..n})}{\mu(x_{1..n})} > n \right) \leq \frac{1}{n} E_\mu \frac{\rho(x_{1..n})}{\mu(x_{1..n})} = \frac{1}{n}. \quad (12)$$

Next, observe that for every $n \in \mathbb{N}$ and every set $A \subset \mathcal{X}^n$, using Jensen’s inequality we can obtain

$$\begin{aligned} - \sum_{x_{1..n} \in A} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} &= -\mu(A) \sum_{x_{1..n} \in A} \frac{1}{\mu(A)} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} \\ &\geq -\mu(A) \log \frac{\rho(A)}{\mu(A)} \geq -\mu(A) \log \rho(A) - \frac{1}{2}. \end{aligned} \quad (13)$$

Moreover,

$$\begin{aligned} d_n(\mu, p(\mu)) &= - \sum_{x_{1..n} \in \mathcal{X}^n \setminus V_\mu^n} \mu(x_{1..n}) \log \frac{p(\mu)(x_{1..n})}{\mu(x_{1..n})} \\ &\quad - \sum_{x_{1..n} \in V_\mu^n} \mu(x_{1..n}) \log \frac{p(\mu)(x_{1..n})}{\mu(x_{1..n})} \geq \delta_n(\mu) \mu(\mathcal{X}^n \setminus V_\mu^n) - 1/2, \end{aligned}$$

where in the inequality we have used (10) for the first summand and (13) for the second. Thus,

$$\mu(\mathcal{X}^n \setminus V_\mu^n) \leq \frac{d_n(\mu, p(\mu)) + 1/2}{\delta_n(\mu)} = o(1). \quad (14)$$

From (11), (12) and (14) we conclude

$$\mu(\mathcal{X}^n \setminus T_\mu^n) \leq \mu(\mathcal{X}^n \setminus V_\mu^n) + \mu(\mathcal{X}^n \setminus U_\mu^n) = o(1). \quad (15)$$

Step 2n: a countable cover, time n. Fix an $n \in \mathbb{N}$. Define $m_1^n := \max_{\mu \in \mathcal{C}} \rho(T_\mu^n)$ (since \mathcal{X}^n are finite all suprema are reached). Find any μ_1^n such that $\rho_1^n(T_{\mu_1^n}^n) = m_1^n$ and let $T_1^n := T_{\mu_1^n}^n$. For $k > 1$, let $m_k^n := \max_{\mu \in \mathcal{C}} \rho(T_\mu^n \setminus T_{\mu_k^n}^n)$. If $m_k^n > 0$, let μ_k^n be any $\mu \in \mathcal{C}$ such that $\rho(T_{\mu_k^n}^n \setminus T_{\mu_k^n}^n) = m_k^n$, and let $T_k^n := T_{\mu_k^n}^n \cup T_{\mu_k^n}^n$; otherwise let $T_k^n := T_{\mu_{k-1}}^n$. Observe that (for each n) there is only a finite number of positive m_k^n , since the set \mathcal{X}^n is finite; let K_n be the largest index k such that $m_k^n > 0$. Let

$$\nu_n := \sum_{k=1}^{K_n} w_k p(\mu_k^n). \quad (16)$$

As a result of this construction, for every $n \in \mathbb{N}$ every $k \leq K_n$ and every $x_{1..n} \in T_k^n$ using the definitions (11), (9) and (10) we obtain

$$\nu_n(x_{1..n}) \geq w_k \frac{1}{n} 2^{-\delta_n(\mu)} \rho(x_{1..n}). \quad (17)$$

Step 2: the resulting predictor. Finally, define

$$\nu := \frac{1}{2}\gamma + \frac{1}{2} \sum_{n \in \mathbb{N}} w_n \nu_n, \quad (18)$$

where γ is the i.i.d. measure with equal probabilities of all $x \in \mathcal{X}$ (that is, $\gamma(x_{1..n}) = |\mathcal{X}|^{-n}$ for every $n \in \mathbb{N}$ and every $x_{1..n} \in \mathcal{X}^n$). We will show that ν predicts every $\mu \in \mathcal{C}^+$, and then in the end of the proof (Step r) we will show how to replace γ by a combination of a countable set of elements of \mathcal{C} (in fact, γ is just a regularizer which ensures that ν -probability of any word is never too close to 0).

Step 3: ν predicts every $\mu \in \mathcal{C}^+$. Fix any $\mu \in \mathcal{C}^+$. Introduce the parameters $\varepsilon_\mu^n \in (0, 1)$, $n \in \mathbb{N}$, to be defined later, and let $j_\mu^n := 1/\varepsilon_\mu^n$. Observe that $\rho(T_k^n \setminus T_{k-1}^n) \geq \rho(T_{k+1}^n \setminus T_k^n)$, for any $k > 1$ and any $n \in \mathbb{N}$, by definition of these sets. Since the sets $T_k^n \setminus T_{k-1}^n$, $k \in \mathbb{N}$ are disjoint, we obtain $\rho(T_k^n \setminus T_{k-1}^n) \leq 1/k$. Hence, $\rho(T_\mu^n \setminus T_j^n) \leq \varepsilon_\mu^n$ for some $j \leq j_\mu^n$, since otherwise $m_j^n = \max_{\mu \in \mathcal{C}} \rho(T_\mu^n \setminus T_j^n) > \varepsilon_\mu^n$ so that $\rho(T_{j_\mu^n+1}^n \setminus T_{j_\mu^n}^n) > \varepsilon_\mu^n = 1/j_\mu^n$, which is a contradiction. Thus,

$$\rho(T_\mu^n \setminus T_{j_\mu^n}^n) \leq \varepsilon_\mu^n. \quad (19)$$

We can upper-bound $\mu(T_\mu^n \setminus T_{j_\mu^n}^n)$ as follows. First, observe that

$$\begin{aligned} d_n(\mu, \rho) &= - \sum_{x_{1..n} \in T_\mu^n \cap T_{j_\mu^n}^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} \\ &\quad - \sum_{x_{1..n} \in T_\mu^n \setminus T_{j_\mu^n}^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} \\ &\quad - \sum_{x_{1..n} \in \mathcal{X}^n \setminus T_\mu^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} \\ &= I + II + III. \end{aligned} \quad (20)$$

Then, from (11) and (9) we get

$$I \geq -\log n. \quad (21)$$

From (13) and (19) we get

$$II \geq -\mu(T_\mu^n \setminus T_{j_\mu^n}^n) \log \rho(T_\mu^n \setminus T_{j_\mu^n}^n) - 1/2 \geq -\mu(T_\mu^n \setminus T_{j_\mu^n}^n) \log \varepsilon_\mu^n - 1/2. \quad (22)$$

Furthermore,

$$\begin{aligned} III &\geq \sum_{x_{1..n} \in \mathcal{X}^n \setminus T_\mu^n} \mu(x_{1..n}) \log \mu(x_{1..n}) \geq \mu(\mathcal{X}^n \setminus T_\mu^n) \log \frac{\mu(\mathcal{X}^n \setminus T_\mu^n)}{|\mathcal{X}^n \setminus T_\mu^n|} \\ &\geq -\frac{1}{2} - \mu(\mathcal{X}^n \setminus T_\mu^n) n \log |\mathcal{X}|, \end{aligned} \quad (23)$$

where the first inequality is obvious, in the second inequality we have used the fact that entropy is maximized when all events are equiprobable and in the third one we used $|\mathcal{X}^n \setminus T_\mu^n| \leq |\mathcal{X}|^n$. Combining (20) with the bounds (21), (22) and (23) we obtain

$$d_n(\mu, \rho) \geq -\log n - \mu(T_\mu^n \setminus T_{j_\mu^n}^n) \log \varepsilon_\mu^n - 1 - \mu(\mathcal{X}^n \setminus T_\mu^n) n \log |\mathcal{X}|,$$

so that

$$\mu(T_\mu^n \setminus T_{j_\mu^n}^n) \leq \frac{1}{-\log \varepsilon_\mu^n} \left(d_n(\mu, \rho) + \log n + 1 + \mu(\mathcal{X}^n \setminus T_\mu^n) n \log |\mathcal{X}| \right). \quad (24)$$

From the fact that $d_n(\mu, \rho) = o(n)$ and (15) it follows that the term in brackets is $o(n)$, so that we can define the parameters ε_μ^n in such a way that $-\log \varepsilon_\mu^n = o(n)$ while at the same time the bound (24) gives $\mu(T_\mu^n \setminus T_{j_\mu^n}^n) = o(1)$. Fix such a choice of ε_μ^n . Then, using (15), we conclude

$$\mu(\mathcal{X}^n \setminus T_{j_\mu^n}^n) \leq \mu(\mathcal{X}^n \setminus T_\mu^n) + \mu(T_\mu^n \setminus T_{j_\mu^n}^n) = o(1). \quad (25)$$

We proceed with the proof of $d_n(\mu, \nu) = o(n)$. For any $x_{1..n} \in T_{j_\mu^n}^n$ we have

$$\nu(x_{1..n}) \geq \frac{1}{2} w_n \nu_n(x_{1..n}) \geq \frac{1}{2} w_n w_{j_\mu^n} \frac{1}{n} 2^{-\delta_n(\mu)} \rho(x_{1..n}) = \frac{w_n w}{2n} (\varepsilon_\mu^n)^2 2^{-\delta_n(\mu)} \rho(x_{1..n}), \quad (26)$$

where the first inequality follows from (18), the second from (17), and in the equality we have used $w_{j_\mu^n} = w/(j_\mu^n)^2$ and $j_\mu^n = 1/\varepsilon_\mu^n$. Next we use the decomposition

$$d_n(\mu, \nu) = - \sum_{x_{1..n} \in T_{j_\mu^n}^n} \mu(x_{1..n}) \log \frac{\nu(x_{1..n})}{\mu(x_{1..n})} - \sum_{x_{1..n} \in \mathcal{X}^n \setminus T_{j_\mu^n}^n} \mu(x_{1..n}) \log \frac{\nu(x_{1..n})}{\mu(x_{1..n})} = I + II. \quad (27)$$

From (26) we find

$$\begin{aligned} I &\leq -\log \left(\frac{w_n w}{2n} (\varepsilon_\mu^n)^2 2^{-\delta_n(\mu)} \right) - \sum_{x_{1..n} \in T_{j_\mu^n}^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} \\ &= (1 + 3 \log n - 2 \log \varepsilon_\mu^n - 2 \log w + \delta_n(\mu)) + \left(d_n(\mu, \rho) + \sum_{x_{1..n} \in \mathcal{X}^n \setminus T_{j_\mu^n}^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} \right) \\ &\leq o(n) - \sum_{x_{1..n} \in \mathcal{X}^n \setminus T_{j_\mu^n}^n} \mu(x_{1..n}) \log \mu(x_{1..n}) \\ &\leq o(n) + \mu(\mathcal{X}^n \setminus T_{j_\mu^n}^n) n \log |\mathcal{X}| = o(n), \quad (28) \end{aligned}$$

where in the second inequality we have used $-\log \varepsilon_\mu^n = o(n)$, $d_n(\mu, \rho) = o(n)$ and $\delta_n(\mu) = o(n)$, in the last inequality we have again used the fact that the entropy is maximized when all events are equiprobable, while the last equality follows from (25). Moreover, from (18) we find

$$II \leq \log 2 - \sum_{x_{1..n} \in \mathcal{X}^n \setminus T_{j_\mu^n}^n} \mu(x_{1..n}) \log \frac{\gamma(x_{1..n})}{\mu(x_{1..n})} \leq 1 + n \mu(\mathcal{X}^n \setminus T_{j_\mu^n}^n) \log |\mathcal{X}| = o(n), \quad (29)$$

where in the last inequality we have used $\gamma(x_{1..n}) = |\mathcal{X}|^{-n}$ and $\mu(x_{1..n}) \leq 1$, and the last equality follows from (25).

From (27), (28) and (29) we conclude $\frac{1}{n} d_n(\nu, \mu) \rightarrow 0$.

Step r: the regularizer γ . It remains to show that the i.i.d. regularizer γ in the definition of ν (18), can be replaced by a convex combination of a countably many elements from \mathcal{C} . Indeed, for each $n \in \mathbb{N}$, denote

$$A_n := \{x_{1..n} \in \mathcal{X}^n : \exists \mu \in \mathcal{C} \mu(x_{1..n}) \neq 0\},$$

and let for each $x_{1..n} \in \mathcal{X}^n$ the measure $\mu_{x_{1..n}}$ be any measure from \mathcal{C} such that $\mu_{x_{1..n}}(x_{1..n}) \geq \frac{1}{2} \sup_{\mu \in \mathcal{C}} \mu(x_{1..n})$. Define

$$\gamma'_n(x'_{1..n}) := \frac{1}{|A_n|} \sum_{x_{1..n} \in A_n} \mu_{x_{1..n}}(x'_{1..n}),$$

for each $x'_{1..n} \in A^n$, $n \in \mathbb{N}$, and let $\gamma' := \sum_{k \in \mathbb{N}} w_k \gamma'_k$. For every $\mu \in \mathcal{C}$ we have

$$\gamma'(x_{1..n}) \geq w_n |A_n|^{-1} \mu_{x_{1..n}}(x_{1..n}) \geq \frac{1}{2} w_n |\mathcal{X}|^{-n} \mu(x_{1..n})$$

for every $n \in \mathbb{N}$ and every $x_{1..n} \in A_n$, which clearly suffices to establish the bound $II = o(n)$ as in (29). \blacksquare

Acknowledgements

This work has been partially supported by the French Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council and FEDER through the ‘‘Contrat de Projets Etat Region (CPER) 2007-2013’’, by the French National Research Agency (ANR), project EXPLO-RA ANR-08-COSI-004, and by Pascal2.

References

- [Blackwell and Dubins, 1962] Blackwell, D. and Dubins, L. (1962). Merging of opinions with increasing information. *Annals of Mathematical Statistics*, 33:882–887.
- [Cesa-Bianchi and Lugosi, 1999] Cesa-Bianchi, N. and Lugosi, G. (1999). On prediction of individual sequences. *Annals of Statistics*, 27:1865–1895.
- [Cesa-Bianchi and Lugosi, 2006] Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, Learning, and Games*. Cambridge University Press.
- [Kalai and Lehrer, 1994] Kalai, E. and Lehrer, E. (1994). Weak and strong merging of opinions. *Journal of Mathematical Economics*, 23:73–86.
- [Krichevsky, 1993] Krichevsky, R. (1993). *Universal Compression and Retrieval*. Kluwer Academic Publishers.
- [Plesner and Rokhlin, 1946] Plesner, A. and Rokhlin, V. (1946). Spectral theory of linear operators, II. *Uspekhi Matematicheskikh Nauk*, 1:71–191.
- [Ryabko, 1984] Ryabko, B. (1984). Twice-universal coding. *Problems of Information Transmission*, 3:173–177.
- [Ryabko, 1988] Ryabko, B. (1988). Prediction of random sequences and universal coding. *Problems of Information Transmission*, 24:87–96.
- [Ryabko, 2009] Ryabko, D. (2009). Characterizing predictable classes of processes. In J. Bilmes, A. N., editor, *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09)*, Montreal, Canada.
- [Ryabko, 2010] Ryabko, D. (2010). On finding predictors for arbitrary families of processes. *Journal of Machine Learning Research*, 11:581–602.
- [Ryabko and Hutter, 2007] Ryabko, D. and Hutter, M. (2007). On sequence prediction for arbitrary measures. In *Proc. 2007 IEEE International Symposium on Information Theory*, pages 2346–2350, Nice, France. IEEE.
- [Ryabko and Hutter, 2008] Ryabko, D. and Hutter, M. (2008). Predicting non-stationary processes. *Applied Mathematics Letters*, 21(5):477–482.
- [Shiryayev, 1996] Shiryaev, A. N. (1996). *Probability*. Springer.
- [Solomonoff, 1978] Solomonoff, R. J. (1978). Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Information Theory*, IT-24:422–432.
- [Ziv and Lempel, 1978] Ziv, J. and Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24:530–536.

Regret Minimization for Online Buffering Problems Using the Weighted Majority Algorithm*

Sascha Geulen
Dept. of Computer Science,
RWTH Aachen University
sgeulen@cs.rwth-aachen.de

Berthold Vöcking
Dept. of Computer Science,
RWTH Aachen University
voecking@cs.rwth-aachen.de

Melanie Winkler
Dept. of Computer Science,
RWTH Aachen University
winkler@cs.rwth-aachen.de

Abstract

Suppose a decision maker has to purchase a commodity over time with varying prices and demands. In particular, the price per unit might depend on the amount purchased and this price function might vary from step to step. The decision maker has a buffer of bounded size for storing units of the commodity that can be used to satisfy demands at later points in time. We seek for an algorithm deciding at which time to buy which amount of the commodity so as to minimize the cost. This kind of problem arises in many technological and economical settings like, e.g., battery management in hybrid cars and economical caching policies for mobile devices. A simplified but illustrative example is a frugal car driver thinking about at which occasion to buy which amount of gasoline.

Within a regret analysis, we assume that the decision maker can observe the performance of a set of expert strategies over time and synthesizes the observed strategies into a new online algorithm. In particular, we investigate the external regret obtained by the well-known Randomized Weighted Majority algorithm applied to our problem. We show that this algorithm does not achieve a reasonable regret bound if its random choices are independent from step to step, that is, the regret for T steps is $\Omega(T)$. However, one can achieve regret $O(\sqrt{T})$ when introducing dependencies in order to reduce the number of changes between the chosen experts. If the price functions satisfy a convexity condition then one can even derive a deterministic variant of this algorithm achieving regret $O(\sqrt{T})$.

Our more detailed bounds on the regret depend on the buffer size and the number of available experts. The upper bounds are complemented by a matching lower bound on the best possible external regret.

*Supported by the DFG GK/1298 “AlgoSyn” and UMIC Research Center

1 Introduction

We study online buffering problems dealing with the management of a storage or buffer for a commodity with varying prices and demands. Our setting is similar to the standard model for regret minimization. In particular, we assume that there is a number of experts corresponding to different strategies for managing the buffer. An online learning algorithm observes the performance of the experts and combines their policies with the objective to achieve a performance close to the performance of the best expert. The buffer makes the problem different from the standard setup in online learning since the algorithm now has a state. Switching between experts means switching between states and this might be costly. Indeed, this switching cost is the major challenge in applying known learning algorithms like the algorithms *Randomized Weighted Majority (RWM)* by Littlestone and Warmuth [16] and *Follow the Perturbed Leader (FPL)* by Kalai and Vempala [14] as these algorithms switch between experts frequently.

Informally, the online problems under consideration can be described as follows. A decision maker has to purchase a commodity over time. Time proceeds in discrete steps. In every step, the decision maker needs to satisfy a demand depending on environmental conditions that are not under control of the decision maker. A price function that might vary from step to step describes at which cost the decision maker can buy which amount of the commodity. The price per unit might be constant or depend on the amount purchased. The decision maker has a buffer of bounded size for storing units of the commodity that can be used to satisfy demands at later points in time. We seek for an algorithm deciding at which time to buy which amount of the commodity so as to minimize the cost.

An illustrative example of a buffering problem is a frugal car driver thinking about at which occasion to buy which amount of gasoline. In this example the price per unit varies over time, but can be assumed to be constant in every step, as typically the price for gasoline at a gas station does not depend on the amount that is bought by a single driver. Other examples are battery management in hybrid cars or economical caching policies for mobile devices. In these examples, the prices typically depend on the amount that is generated or purchased. For some applications like the battery management, the price functions may satisfy certain convexity assumptions. Some more information about these applications is given in Section 1.5.

Englert *et al.* [7] study online buffering problems within the framework of competitive analysis. They introduce the economical caching problem and give an online algorithm achieving the best possible competitive ratio against input sequences generated by an adversary. Although this work settles the problem in the competitive framework, the "optimal online algorithm" does not seem to be practical since it acts extremely precariously in order to ensure a worst-case guarantee against the adversary. In fact, the design principle underlying this algorithm is called "thread-based" meaning that the algorithm, in each step, buys the minimal amount that is needed to ensure the competitiveness for all possible extensions of the price sequence. We believe that this kind of risk-averse behavior does not reflect the speculative nature of economical decision makers who may take the risk of buying more units than necessary when speculating on rising prices.

In this paper, we take a less pessimistic approach. As in the competitive framework, the online learning algorithm itself does not have any information about future prices. However, we assume that the algorithm can observe the performance of some experts (online strategies) in preceding steps. Each of these experts may have certain assumptions or knowledge about future prices. For example, one of these experts may assume that prices are set by an adversary and apply the "optimal online algorithm" from [7]. Other experts, may use stochastic prediction rules for estimating future prices [13, 8] or place their decisions based on practical experience and heuristics [5, 15]. Regardless of how these experts are chosen, the objective of the online learning algorithm is to come close to the performance of the best expert.

Before presenting our results, let us formally introduce the traditional model for expert based online learning and the adaption of this model for online buffering.

1.1 Online Learning and Weighted Majority Algorithm

In standard online learning the decision maker is equipped with N experts, numbered from 1 to N . The setup with respect to the cost is different from ours. (In particular, there is no buffer.) One might assume that an adversary chooses the cost for each expert in each step arbitrarily from $[0, 1]$. For expert i , we denote by c_i^t its cost in time step t and by $C_i^t = \sum_{k=1}^t c_i^k$ its accumulated cost until step t . In every step t , the decision maker selects an expert. The cost of the decision maker corresponds to the cost of the expert chosen in that step. Afterwards, it observes the cost vector $c^t \in [0, 1]^N$ of the experts.

Algorithm 1 (Randomized Weighted Majority (RWM))

- 1: $w_i^1 = 1, q_i^1 = \frac{1}{N}$, for all i
 - 2: **for** $t = 1, \dots, T$ **do**
 - 3: choose expert e^t at random according to $Q^t = (q_1^t, \dots, q_N^t)$
 - 4: $w_i^{t+1} = w_i^t(1 - \eta)c_i^t$, for all i
 - 5: $q_i^{t+1} = \frac{w_i^{t+1}}{\sum_{j=1}^N w_j^{t+1}}$, for all i
 - 6: **end for**
-

The decision maker aims at choosing the experts in such a way that its cost is close to the cost of the best expert, that is, it aims at minimizing its regret. The decision maker corresponds to a (possibly randomized) algorithm \mathcal{A} . Consider a sequence of length T . Let $C_{\mathcal{A}}^T$ denote the expected cost accumulated by \mathcal{A} until step T . Formally, the (*external*) *regret* of \mathcal{A} on this sequence is

$$C_{\mathcal{A}}^T - C_{\text{best}}^T .$$

Observe that there are no assumptions on the quality of the experts or the relation between the experts. In general, if the decisions of each expert are arbitrarily bad, online learning algorithms cannot achieve good solutions. In particular, regret minimization does not mean to be competitive to an optimal offline algorithm, as in competitive analysis [3, 17]. However, one of the experts might use a strategy guaranteeing a competitive ratio against adversarial input sequences in which case the online learning algorithm can give (almost) the same performance guarantee as it achieves (almost) the performance of the best expert.

The *Randomized Weighted Majority (RWM)* algorithm of Littlestone and Warmuth [16] guarantees a regret bound of $O(\sqrt{T \log N})$, see also [2]. It is known that this is the best possible bound in the standard setting. The idea of this algorithm is to give each expert a probability of being chosen which depends on the cost that the expert has experienced in the past. The probability q_i that the strategy of expert i is chosen in the next time step is controlled by the current weight w_i of the expert which itself depends only on the weights in the steps up to $t - 1$, the cost c_i^t and a parameter $\eta \in [0, \frac{1}{2}]$. The calculation of the weights and of the probabilities used by the algorithm is described in Algorithm 1. In the rest of the paper the probability for choosing expert i in round t is denoted by q_i^t , the corresponding weight is denoted by w_i^t and η is a parameter from $[0, \frac{1}{2}]$.

1.2 Extending the Online Learning Model towards the Buffering Problem

We study the following online buffering problem. A decision maker has to purchase a commodity over time. Time proceeds in discrete steps. In step t , the decision maker needs to satisfy a demand of $d^t \in [0, 1]$ units of the commodity. The decision maker has a buffer of bounded size $B > 0$ for storing units of the commodity that can be used to satisfy demands at later points in time. In step t , it can purchase at most b^t units, where $b^t \in [d^t, B + d^t]$. The price per unit of the commodity varies over time and depends on the amount bought by the decision maker. It is described by a function $p^t : [0, b^t] \rightarrow [0, 1]$. So the price for buying x units in step t is given by $x p^t(x)$. We seek for an algorithm deciding at which time to buy which amount of the commodity so as to minimize the cost.

In the context of the buffering problem, we assume that there are N experts corresponding to online algorithms and each expert is equipped with a buffer of size B . The expert decides how many units to buy in which step. Recall that the price per unit in step t depends on the purchased amount and is defined by the price function p^t . Observe that the experts may buy up to $B + 1$ units per step so that the total price for the purchased units can be as high as $B + 1$.

In our analysis, we account for the purchased units not at the time when the expert (or the online learning algorithm) buys the units but when it uses them to satisfy the demand. To formalize this, assume that every amount purchased by the expert (or the online algorithm) is put into the buffer and all demands are satisfied from the buffer in first-in first-out (FIFO) manner. The cost accounted for satisfying a demand with units bought in previous steps is equal to the price at which the units were bought. This accounting trick ensures that the cost of the experts (and the online algorithm) is at most one per step and it only decreases the accumulated cost of the experts up to an additive value of at most B . The cost by expert i in time step t is termed c_i^t , its cost accumulated until step t is denoted by $C_i^t = \sum_{k=1}^t c_i^k$ like in the standard model.

In every step, an online learning algorithm \mathcal{A} chooses (possibly at random) one of the experts (or a linear combination of experts). In step t this choice depends only on the demands and prices until

step $t - 1$, i.e., on $d^1, \dots, d^{t-1}, p^1, \dots, p^{t-1}$. The term *choosing an expert* needs further clarification. If \mathcal{A} has chosen expert i in step t , then it purchases the same amount of the commodity as expert i in step t with the following two exceptions: If the amount purchased by the expert together with the units in \mathcal{A} 's buffer does not suffice to cover the demand in this step, then \mathcal{A} purchases a minimal amount necessary to satisfy the demand. (After such a step the buffer is empty.) If the amount purchased by the expert exceeds the demand in this step and the excess does not fit completely into \mathcal{A} 's buffer, then it purchases only the amount needed to fill the buffer up to the capacity. (After such a step the buffer contains B units.)

The (expected) cost of algorithm \mathcal{A} in step t is termed $c_{\mathcal{A}}^t$, and the cost that \mathcal{A} accumulates until step t is denoted by $C_{\mathcal{A}}^t = \sum_{k=1}^t c_{\mathcal{A}}^k$.

1.3 Our Contribution

We investigate the regret achieved by the RWM algorithm and variations of this algorithm on the buffering problem.

When describing the RWM algorithm, we did not specify that the random experiments in different steps are independent. In fact, in the standard setting such dependencies do not effect the expected cost of the RWM algorithm.¹ This is different when applying the algorithm to the buffering problem. In particular, one does not obtain a reasonable bound on the regret if experts are chosen using an independent random experiment for every step.

To see this, consider the following input sequence with fixed per unit prices (i.e., the price functions p^t are constant):

$$\begin{bmatrix} p^t \\ d^t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 1/4 \end{bmatrix} \begin{bmatrix} 1 \\ 1/4 \end{bmatrix} \begin{bmatrix} 0 \\ 1/4 \end{bmatrix} \begin{bmatrix} 1 \\ 1/4 \end{bmatrix} \right)^{T'}$$

The sequence consists of an *initial step* with cost and demand equal to 0 followed by T' rounds of four steps with cost 0 or 1 (as specified above) and demand 1/4 each. The length of the sequence is $T = 4T' + 1$. Suppose the buffer size is 1. The following two experts are given:

- The first expert purchases 1/2 unit in the initial step and afterwards one unit in the third step of every round.
- The second expert purchases one unit in the first step of every round.

Obviously, both experts have cost 0 for the whole request sequence. We claim, however, that RWM has cost $\Theta(T)$ and, hence, the regret is $\Theta(T)$, too. RWM assigns probability 1/2 to each of the experts in each step. If random experiments in different steps are independent, then in two consecutive rounds, with probability 1/16 the algorithm selects in each step an expert that does not purchase a unit in this step. In this case, the buffer of RWM is empty in each step of the second round, which means that RWM has cost 1/2 for serving the demands in the second round. Thus, for each round the expected cost of RWM is at least 1/32.

A reason for the poor performance of RWM is that it changes the experts frequently and the chosen experts might have completely different filling levels in their buffers. We present a variant of RWM that uses dependencies in order to decrease the number of expert changes. The algorithm is called *Shrinking Dartboard (SD)* algorithm as the random experiments used by this algorithm are described in terms of a shrinking dartboard. We prove that the regret achieved by this algorithm is at most $O(\sqrt{BT \log N})$.

If the price functions in each step satisfy a convexity condition (e.g., if prices are constant) then we can derive even a deterministic variant of the Weighted Majority Algorithm with a good regret bound. The *Weighted Fractional (WF)* algorithm chooses linear combinations of experts rather than selecting experts at random. WF achieves a regret of at most $O(\sqrt{BT \log N})$, too. As this algorithm is deterministic, its regret guarantee holds even against an adaptive adversary.

Finally, we present a lower bound of $\Omega(\sqrt{BT \log N})$ for the regret showing that SD and WF achieve the optimal regret for the buffering problem up to constant factors. This lower bound holds even if prices are assumed to be constant.

¹In the standard setting without buffer, the expected cost of the RWM algorithm is not effected by dependencies between different steps, unless the adversary who specifies the cost for the experts is *adaptive*, i.e., the cost vector presented for a step might depend on the random coin flips of the algorithm in previous steps.

1.4 Related Work

A special case of online buffering problems is the *economical caching* [7, 8] and the *one-way trading* problem [3, 6]. In one-way trading a price sequence is given, each price representing an exchange rate from dollar to yen. The task is to trade d dollars to a maximum number of yen. In economical caching, there is furthermore a previously unknown demand of yen at each time step which must be consumed from a buffer or bought for the current price. Economical caching was introduced in [7]. Englert et al. analyzed it in a worst case competitive analysis yielding general tight bounds for the competitive factor depending on the ratio between the lowest and highest exchange rate. [8] showed that if the price sequence is modeled by a random walk, a competitive factor which does not depend on this ratio can be achieved.

We study online buffering problems with online learning algorithms as described in Section 1.2. Online learning algorithms are used for regret minimization. A general introduction is given in [1, 2]. In this paper we use the external regret model [2, 4] with full information.

There are several algorithms for which the regret per time step converges against zero for a long time horizon. The following algorithms achieve a regret of $O(\sqrt{T \log N})$: The *Randomized Weighted Majority* (RWM) algorithm [16] weights each expert depending on its cost so far. In the randomized version the weights are used as probabilities for an expert to be chosen. It is known that RWM achieves the best possible bound for the standard setting. But this is no longer valid when applying the algorithm to the online buffering problem. A reason for this is that RWM changes the experts frequently.

[12] and [14] present the *Follow The Perturbed Leader* (FPL) algorithm that follows the expert which has achieved the lowest cost so far plus some perturbation. FPL reduces the number of expert changes to achieve good regret bounds. But in contrast to the Weighted Fractional (WF) algorithm presented in this paper FPL cannot achieve this bound against an adaptive adversary when applied to online buffering problems.

The WF algorithm can only achieve good regret bounds if the price functions satisfy a convexity property. This assumption is also made in [9, 18]. It is shown that if the cost of each expert is given by a convex function which may change over time, a gradient descent algorithm can be used to achieve good regret bounds. This algorithm cannot directly be used to achieve good regret bounds for online buffering problems, since the experts choose a fixed point of the convex function. In our model this would lead to very limited experts. An expert would then only be able to determine once the number units it wants to buy in every time step. This choice would be the same for all time steps. It could no longer depend on the current price or filling status of the buffer. This expert model is too limited and can therefore not be used to solve online buffering problems.

Online learning algorithms have been studied in many other research areas yielding a huge variety of results. Some of the possible applications for online learning are given in [10].

1.5 Applications

One possible application for our online learning algorithms is *battery management* for hybrid cars. In a hybrid car there are two engines, a combustion engine and an electrical engine. The energy for the electrical engine is taken from a battery of bounded capacity. The battery can be recharged using the combustion engine or regenerative energy, e.g., from the braking system. The torque is provided by both of these engines and the demand of torque depends on the acceleration requested by the driver and the topology of the route.

The online algorithm has to decide how much power is produced by each of the two engines. If more power is required than the combustion engine is producing, the remaining power must be used from the battery or it must be produced by the electrical engine by using regenerative energies. On the other hand, if more power is produced than currently needed, the additional power is not given to the drive shaft, but saved into the battery for later usage.

In our model, the demand specified in the input sequence corresponds to the energy needed for providing the requested torque as well as for electrical devices, e.g., for air conditioning. The amount that is purchased by the online algorithm corresponds to the energy produced by the combustion engine and the exploited regenerative energy. The price of the energy generated by the combustion engine is determined by the used amount of fuel, which itself depends on the torque and the gear. These price functions potentially satisfy the convexity assumption used in the analysis of the WF algorithm. The price of the exploited regenerative energy is 0.

In engineering, decisions about which engine should produce which amount of power are typically made based on engine operating maps [5] and certain knowledge of the route [13, 15]. Engine operating maps show the fuel consumption of the car for different operation states. The topology of the route can, e.g., be estimated by using the on-board navigation system. Heuristics based

Algorithm 2 (Shrinking Dartboard (SD))

- 1: $w_i^1 = 1, q_i^1 = \frac{1}{N}$, for all i
 - 2: choose expert e^1 at random according to $Q^1 = (q_1^1, \dots, q_N^1)$
 - 3: **for** $t = 2, \dots, T$ **do**
 - 4: $w_i^t = w_i^{t-1}(1 - \eta)^{c_i^{t-1}}$, for all i
 - 5: $q_i^t = \frac{w_i^t}{\sum_{j=1}^N w_j^t}$, for all i
 - 6: with probability $\frac{w_{e^t}^t}{w_{e^t}^{t-1}}$ do not change expert, i.e., set $e^t = e^{t-1}$
 - 7: else choose e^t at random according to $Q^t = (q_1^t, \dots, q_N^t)$
 - 8: **end for**
-

on engine operating maps parametrized with typical driving conditions combined with different prediction models for the topology are suitable candidates for the experts used by our online learning algorithms.

Another application in a completely different context is *smart caching* of data streams on mobile devices, see also [7, 11]. Suppose a data stream can be fetched by a mobile device over different communication standards like, e.g., GSM, UMTS, WLAN, each for different cost, but not all services are always and anywhere available. The stream can be buffered by the mobile device in a storage of bounded size. We assume that the most expensive of these services is always available at a cost of one per data unit and any demand per step can always be satisfied using this expensive service. Other services can be used to download and buffer the data stream at lower cost only if they are available.

The best way for combining the different services depends on the users mobility and the availability of the services over time. Therefore, it is not possible to implement a fixed optimal strategy into the mobile device, but a good strategy shall be learned online. The experts for online learning in smart caching recommend which standard to use at which time step. They might recommend also to combine the different communication standards by using each standard to download a certain amount of the data in a time step.

In this context it might not be appropriate to assume that the price function is convex. Besides it might not be possible that every arbitrary amount of data can be downloaded per time step, but some services are restricted to a certain variety of different amounts. Under these assumptions, our algorithm WF cannot be applied as it combines experts in a fractional manner and its analysis assumes convexity of the price function. Let us remark, however, that SD can be applied without any assumptions on the price functions and even if different services are restricted to discrete amounts of data as long as the simulated experts satisfy the restrictions regarding the amounts that can be downloaded.

2 The Shrinking Dartboard Algorithm

We devise a variant of the Randomized Weighted Majority algorithm using dependencies between the random decisions in different steps in order to reduce the number of expert changes. Algorithm 2 specifies how the experts are chosen and introduces the notation for this section. Furthermore, let $W^t = \sum_{i=1}^N w_i^t$ denote the sum of weights in step t . The algorithm is called *Shrinking Dartboard (SD)* algorithm as it can be illustrated in terms of a dartboard shrinking over time.

Initially, the dartboard is a disc divided into N equally sized sectors, one for each expert. The total area covered by the disc has size N so that each sector has size 1. In step 1, SD chooses an expert by *throwing a dart* to the board, that is, it picks a point from the disc uniformly at random and chooses that expert into which sector this point falls. Over time, the expert's sectors shrink as illustrated in Figure 1. In particular, the size of the area covered by expert i 's sector in step t , denoted by *allowed area* in step t , corresponds to the weight w_i^t as specified in Algorithm 2. In step $t > 1$, SD chooses an expert as follows: If the previously picked point is still in the allowed area, then SD does not change the expert. This happens with probability $w_{e^t}^t/w_{e^t}^{t-1}$ and corresponds to line 6 of Algorithm 2. Otherwise, SD *throws a new dart*, that is, it picks a point uniformly at random from the area covered by the sectors of all experts and chooses the expert into which sector this point falls. This happens with probability $1 - w_{e^t}^t/w_{e^t}^{t-1}$ and corresponds to line 7 of Algorithm 2.

Observe that the weights used by SD correspond to the weights of the original Randomized Weighted Majority algorithm (Algorithm 1). The following lemma shows that SD does not only use the same weights as RWM but both algorithms have the same probability distribution for choosing

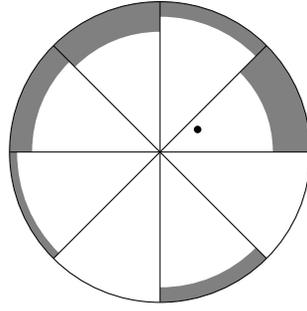


Figure 1: Probability distribution as a dartboard

an expert in a step. The difference, however, is that the random choices made by SD in different steps are not independent, but the selection of the expert in step t depends on the selection in step $t - 1$.

Lemma 1 $\Pr[e^t = i] = q_i^t$, for $i \in \{1, \dots, N\}$, $t \in \{1, \dots, T\}$.

Proof: We use an induction on $1 \leq t \leq T$. For $t = 1$, the statement in the lemma follows immediately from the description of the algorithm. Now let $t \geq 2$. Expert i is selected in step t either because it was selected already in step $t - 1$ and the corresponding dart is still in the allowed area (i.e., the expert is chosen by line 6 of Algorithm 2) or because a new dart is thrown and this dart hits i 's sector (i.e., the expert is chosen by line 7). Hence,

$$\begin{aligned}
\Pr[e^t = i] &= \Pr[e^{t-1} = i] \cdot \frac{w_i^t}{w_i^{t-1}} + q_i^t \cdot \sum_{j=1}^N \Pr[e^{t-1} = j] \cdot \left(1 - \frac{w_j^t}{w_j^{t-1}}\right) \\
&= q_i^{t-1} \cdot \frac{w_i^t}{w_i^{t-1}} + q_i^t \cdot \sum_{j=1}^N q_j^{t-1} \cdot \left(1 - \frac{w_j^t}{w_j^{t-1}}\right) \\
&= \frac{w_i^{t-1}}{W^{t-1}} \cdot \frac{w_i^t}{w_i^{t-1}} + \frac{w_i^t}{W^t} \cdot \sum_{j=1}^N \frac{w_j^{t-1}}{W^{t-1}} \cdot \frac{w_j^{t-1} - w_j^t}{w_j^{t-1}} \\
&= \frac{w_i^t}{W^{t-1}} + \frac{w_i^t}{W^t} \cdot \frac{W^{t-1} - W^t}{W^{t-1}} = \frac{w_i^t}{W^t} = q_i^t.
\end{aligned}$$

■

Let D denote the number of expert changes during the execution of SD. The following lemma bounds the expected value of D in terms of the cost of the best expert.

Lemma 2 $\mathbf{E}[D] \leq 2\eta C_{\text{best}}^T + \ln N$.

Proof: D is bounded from above by the number of times line 7 of the algorithm is applied which corresponds to the number of darts that are thrown because the sector of the chosen expert shrinks. The probability for throwing a new dart in step $t \geq 2$ is

$$\alpha^t = \sum_{j=1}^N \Pr[e^{t-1} = j] \cdot \left(1 - \frac{w_j^t}{w_j^{t-1}}\right) = \frac{W^{t-1} - W^t}{W^{t-1}},$$

where the latter equation follows from the calculation in the proof of Lemma 1. Thus, $W^t = (1 - \alpha^t)W^{t-1}$.

The total weight W^{T+1} after step T can hence be expressed in terms of these probabilities, that is,

$$W^{T+1} = W^1 \prod_{t=1}^T (1 - \alpha^{t+1}) = N \prod_{t=1}^T (1 - \alpha^{t+1}).$$

On the other hand,

$$W^{T+1} \geq (1 - \eta) C_{\text{best}}^T$$

as the latter quantity corresponds to the weight of the best expert after step T . Combining these equations and applying the logarithm gives

$$C_{\text{best}}^T \ln(1 - \eta) \leq \ln N + \sum_{t=1}^T \ln(1 - \alpha^{t+1}) .$$

Using $\ln(1 - \alpha^{t+1}) \leq -\alpha^{t+1}$ and $\ln(1 - \eta) \geq -2\eta$, the expected number of thrown darts is thus

$$\sum_{t=1}^{T-1} \alpha^{t+1} \leq \ln N + 2\eta C_{\text{best}}^T .$$

■

We apply the lemmas above in the following regret analysis in which we compare the cost of SD with the cost of the best expert. As defined in the Section 1.2, C_i^t is the cost of the units expert i uses (rather than purchases) until step t , where we assume that units are consumed from the expert's buffer in FIFO manner and valued with the price at which they were bought. Recall that c_i^t denotes the cost accounted for expert i in step t .

Theorem 3 *For $\eta \leq 1/2$, the expected cost of SD satisfies*

$$C_{\text{SD}}^T \leq (1 + \eta + 2\eta B)C_{\text{best}}^T + \frac{\ln N}{\eta} + B \ln N .$$

Setting $\eta = \min\{\sqrt{\ln N/(BT)}, 1/2\}$ yields $C_{\text{SD}}^T \leq C_{\text{best}}^T + O(\sqrt{BT \log N})$.

Proof: We claim that the cost of SD is bounded from above by

$$\sum_{t=1}^T c_{e^t}^t + DB .$$

In words, the cost of SD is bounded from above by the cost of the chosen experts plus the number of expert changes times the buffer size. To see this, consider the cost in a time period beginning with a step in which a new expert is chosen and ending with the last step before the next expert is chosen or the sequence ends. The cost of SD in this period can be split into three contributions.

- (1) Cost due to units used and bought in this period by both SD and the expert.
- (2) Cost for using those units that are stored in SD's buffer at the beginning of the period.
- (3) Cost for using units bought during the period by SD to ensure feasibility.

The cost in (1) is upper bounded by $\sum_{t=1}^T c_{e^t}^t$. Observe that the sum of units covered by (2) and (3) is at most B . Hence, the difference between the cost of SD and the cost of the expert within such a period is at most B . Furthermore, in the very first period, the costs in (2) and (3) are 0 because both SD and the expert start with an empty buffer and purchase and use the same units. This gives the stated upper bound on C_{SD}^T as the number of periods without counting the first period is D .

Next we claim that

$$E \left[\sum_{t=1}^T c_{e^t}^t \right] \leq (1 + \eta)C_{\text{best}}^T + \frac{\ln N}{\eta} .$$

This follows from Lemma 1 showing that the probability that SD chooses expert i in step t is equal to the probability that RWM chooses expert i in step t . The left hand term describes the cost of RWM assuming that the cost accounted for the learning algorithm in step t are $c_{e^t}^t$ (as in the standard setting of online learning). Thus, we can apply the well known upper bound on the cost of RWM holding for $\eta \leq 1/2$ (cf., e.g., [2]).

Together with the bound on the expected value of D in Lemma 2 this yields the first bound in the theorem.

Finally, assume $\eta \leq \sqrt{\ln N/(BT)}$. Combining the first bound of the theorem with the trivial bound $C_{\text{SD}}^T - C_{\text{best}}^T \leq T$, we obtain

$$\begin{aligned} C_{\text{SD}}^T - C_{\text{best}}^T &\leq (\eta + 2\eta B)T + \frac{\ln N}{\eta} + \min\{T, B \ln N\} \\ &= O\left(\sqrt{BT \log N} + \min\{T, B \log N\}\right) \\ &= O\left(\sqrt{BT \log N}\right), \end{aligned}$$

which gives the second bound stated in the theorem. ■

3 The Weighted Fractional Algorithm

The *Weighted Fractional (WF)* algorithm uses the same weights as the algorithms RWM and SD. However, instead of choosing an expert at random, it simulates the experts fractionally. That is, it purchases $x^t = \sum_{i=1}^N q_i^t x_i^t$ units in step t , where x_i^t is the amount purchased by expert i in the same step. Observe that this rule might lead to infeasibilities as the amount of purchased units together with the units in the buffer might not be enough to satisfy the demand in a step or the buffer might overflow. In these cases, WF enforces feasibility by purchasing a minimal amount of additional units or reducing the amount of bought units, respectively.

The following theorem bounds the cost of WF assuming that the price functions satisfy a convexity property. In particular, the function $f^t(x) = xp^t(x)$ that describes the cost incurred for buying an amount of x needs to be convex.

Theorem 4 *Suppose the functions $f^t(x)$, $x \in [0, b^t]$ are convex, for $1 \leq t \leq T$. Then the cost of WF satisfies*

$$C_{\text{WF}}^T \leq (1 + \eta + 2\eta B)C_{\text{best}}^T + \frac{\ln N}{\eta} + B \ln N .$$

Setting $\eta = \min\{\sqrt{\ln N/(BT)}, 1/2\}$ yields $C_{\text{WF}}^T \leq C_{\text{best}}^T + O(\sqrt{BT \log N})$.

Proof: We analyze WF by relating it to another algorithm called k -SD. This algorithm splits the buffer into $k \geq 1$ sub-buffers of size B/k each. For each of these sub-buffers, we simulate algorithm SD scaling down all demands as well as the amounts purchased by the experts by multiplying with $1/k$. Besides, we adapt the price function, that is, when buying x_j^t units for sub-buffer j in step t , algorithm k -SD assumes that this incurs *virtual cost* of $f^t(kx_j^t)/k$.

For $1 \leq j \leq k$, let L_j^T denote the expected virtual cost for sub-buffer j accumulated until step T . As k -SD simulates SD for every sub-buffer using an appropriate scaling, it holds $L_j^T = C_{\text{SD}}^T/k$ so that

$$\sum_{j=1}^k L_j^T = C_{\text{SD}}^T .$$

Now let us compare the sum of the virtual cost of k -SD with the true cost of this algorithms. For every time step t , we have

$$f^t \left(\sum_{j=1}^k x_j^t \right) \leq \frac{1}{k} \sum_{j=1}^k f^t(kx_j^t)$$

by Jensen's inequality. Thus, we observe that the true cost incurred for any step t is upper bounded by the sum of the virtual cost for this steps. Combining this observation with the equation above gives

$$C_{k\text{-SD}}^T \leq C_{\text{SD}}^T ,$$

for every $k \geq 1$.

Let us introduce a slight modification to k -SD. The resulting algorithm is called k -SD'. As k -SD simulates SD on each sub-buffer, it needs to buy additional units in some time steps in order to ensure feasibility. These are at most B/k units for every thrown dart (new expert) for each sub-storage. k -SD' does not need to buy these additional units when the respective sub-buffer is empty but can defer this until all sub-buffers are empty. This does not increase the number of units that need to be bought, but prices for these units might change. However, in the analysis of SD, we estimated the prices for these units with the worst possible price. Hence, we can apply Theorem 3 not only to k -SD, but also to k -SD' and obtain

$$C_{k\text{-SD}'}^T \leq (1 + \eta + 2\eta B)C_{\text{best}}^T + \frac{\ln N}{\eta} + B \ln N ,$$

for every $k \geq 1$.

Now we let k go to infinity. Consider a fixed step t . By the law of large numbers, the sum of the amounts purchased by the experts chosen for the sub-buffers converges to its expectation. That is, the sum of purchased amounts over all sub-buffers converges to $\sum_{i=1}^N q_i^t x_i^t$. As a consequence, k -SD' (for $k \rightarrow \infty$) purchases the same amount per step as WF. Hence,

$$C_{\text{WF}}^T = \lim_{k \rightarrow \infty} C_{k\text{-SD}'}^T .$$

Combining the last two equations yields the theorem. ■

4 Lower Bound

The following theorem shows that our upper bounds on the external regret achieved by SD and WF are tight up to constant factors, that is, one cannot achieve significant further improvements. Let us remark that the lower bound given in the theorem holds even if we restrict the input sequences to fixed prices per unit and if the experts purchase at most one unit per step.

Theorem 5 *For every integer T , there exists a stochastically generated sequence of length T together with N experts such that every learning algorithm \mathcal{A} with a buffer of size B suffers a regret of $\Omega(\sqrt{BT \log N})$.*

Proof: The sequence consists of T' consecutive *rounds*. Each round consists of 3 *phases* each of which has B steps so that the sequence has a total length of $T = 3BT'$ steps. Prices are defined by constant functions, i.e., there is a fixed price p^t per unit in every step t . For simplicity in notation, we assume that prices are chosen from the interval $[0,4]$ instead of $[0,1]$. In particular, the sequence of prices and demands has the following structure

$$\begin{bmatrix} p^t \\ d^t \end{bmatrix} = \left(\begin{bmatrix} 2 \\ 0 \end{bmatrix}^B \begin{bmatrix} \{0,4\} \\ 0 \end{bmatrix}^B \begin{bmatrix} 4 \\ 1 \end{bmatrix}^B \right)^{T'}$$

Prices in the second phase of each round are chosen at random from $\{0,4\}$. However, the choices within the same round are dependent, that is, for all steps within the second phase of the same round, we set the same price and this price is chosen uniformly at random from the set $\{0,4\}$. Prices for different rounds are selected independently.

The N experts for the problem are defined as follows: In each round, every expert chooses independently one of the following two strategies each with probability $1/2$.

- a) The expert purchases B units in the first phase.
- b) The expert purchases B units in the second phase.

In both cases, no further units are bought. In particular, the buffer is empty at the end and the beginning of each round.

Let F_i^t denote the random variable defining the cost of expert i in round t . Depending on the outcome of the price for round t and the strategy chosen by the expert, this variable takes the following values.

F_i^t	0	4
a	$2B$	$2B$
b	0	$4B$

When analyzing this random variable, we can assume that, a *column player* (the designer of the sequence) selects a column of this matrix and a *row player* (the considered expert) selects a row. Both of these choices are made independently, uniformly at random. As every entry of the matrix is chosen with probability $1/4$, the expected value of F_i^t is $2B$. Thus, by linearity of expectation, the expected cost of an expert over the whole sequence is $2BT'$.

Next we analyze the expected cost of the *best* expert. Towards this end, let F^t be a random variable describing the average cost in the column chosen by the column player in round t , that is, if the column player chooses column 0 then $F^t = B$ and if the column player chooses column 1 then $F^t = 3B$. The expected value of this variable is $2B$ so that

$$E \left[\sum_{t=1}^{T'} F^t \right] = 2BT' = \frac{2}{3} T .$$

Now define $\Delta_i^t = F_i^t - F^t$. The values taken by the random variable Δ_i^t depend on the choices of the row and column players and are specified in the following matrix.

Δ_i^t	0	4
a	B	$-B$
b	$-B$	B

Observe that the random variables Δ_i^t , $1 \leq i \leq N$, $1 \leq t \leq T'$ are stochastically independent since each of these variables takes one of the value B or $-B$ with probability $1/2$ each, regardless of the outcome of the other variables. For $1 \leq i \leq N$, let $S_i = \sum_{t=1}^{T'} \Delta_i^t/B$. The random variables S_1, \dots, S_N are stochastically independent and each of them corresponds to the value of a fair random walk after T' steps. The expected minimum over N such variables is known to be $-\Theta(\sqrt{T' \log N})$, which gives

$$E \left[\min_i \left(\sum_{t=1}^{T'} \Delta_i^t \right) \right] = -\Theta(B\sqrt{T' \log N}) = -\Theta(\sqrt{BT \log N}) .$$

Hence, the expected cost of the best expert can be estimated by

$$\begin{aligned} E \left[\min_i \left(\sum_{t=1}^{T'} F_i^t \right) \right] &= E \left[\min_i \left(\sum_{t=1}^{T'} (F^t + \Delta_i^t) \right) \right] \\ &= E \left[\sum_{t=1}^{T'} F^t \right] + E \left[\min_i \left(\sum_{t=1}^{T'} \Delta_i^t \right) \right] \\ &= \frac{2}{3}T - \Theta(\sqrt{BT \log N}) . \end{aligned}$$

Finally, we show that any online learning algorithm \mathcal{A} equipped with these experts cannot achieve an expected cost better than $2/3T$. W.l.o.g, \mathcal{A} does not purchase any unit in the third phase of a round, but exactly B units during the first two phases of each round. In the first phase of a round, \mathcal{A} does not have any information about the price in the second phase since the experts decisions (over the whole sequence) and costs (before entering the second phase) do not depend on this price and, hence, do not give any evidence about the price. Thus, \mathcal{A} 's decision about how many of the B units to purchase in the first and how many units to purchase in the second phase of a round is independent of the price selected for the second phase. As a consequence, the expected cost for each purchased unit is 2 and, hence, the expected cost per round is $2B$. Therefore, the expected cost of \mathcal{A} for the whole sequence is $2BT' = 2/3T$ and, consequently, the regret is $\Theta(\sqrt{BT \log N})$. \blacksquare

5 Discussion

We observed that RWM with independent coin flips in different steps fails to give reasonable regret bounds for buffering problems as experts have a state and switching between different states is expensive. We addressed this problem by changing the randomized selection in such a way that changes between experts are reduced. Let us remark that the same kind of problem occurs for algorithm Follow the Perturbed Leader (FPL) by Kalai and Vempala [14], too. The number of expert changes of FPL can be reduced in a similar fashion: One uses only one initial perturbation rather than independent perturbations in every step. In fact, one can deduce from the analysis in [14] that this results in the same regret bound as achieved by our algorithm SD. However, SD has one additional advantage: It can easily be transformed into a simple deterministic, fractional variant of SD guaranteeing optimal regret against an adaptive adversary, provided the price functions satisfy a convexity condition. The same kind of transformation cannot be directly applied to FPL as the probabilities for choosing an expert in a step are not available in closed form for FPL but would need to be extracted from the perturbation experiment.

In the model for online learning with buffer, we have assumed that experts have identical price functions. We want to point out, however, that this assumption is not necessary for our randomized algorithm. It is easy to check that the analysis for SD goes through even if different experts have different price functions within the same step. In contrast, for the deterministic, fractional algorithm WF, the assumption of identical price functions is crucial. It is an interesting question whether one can achieve a similar regret bound for a randomized or deterministic learning algorithm against an adaptive adversary even if experts have different and/or non-convex price functions.

References

- [1] A. Blum. On-line algorithms in machine learning. In *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, chapter 14, pages 306–325. Springer, 1998.
- [2] A. Blum and Y. Mansour. Learning, regret minimization, and equilibria. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 4, pages 79–101. Cambridge University Press, New York, NY, USA, 2007.
- [3] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, NY, USA, 1998.
- [4] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth. How to use expert advice. In *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 382–391, 1993.
- [5] M. Ehsani, Y. Gao, and K. L. Butler. Application of Electrically Peaking Hybrid (ELPH) Propulsion System to a Full-Size Passenger Car with Simulated Design Verification. *IEEE Transactions on Vehicular Technology*, 48(6):1779–1787, 1999.
- [6] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin. Optimal search and one-way trading online algorithms. *Algorithmica*, 30(1):101–139, 2001.
- [7] M. Englert, H. Röglin, J. Spönemann, and B. Vöcking. Economical caching. In *Proceedings of the 26th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 385–396, 2009.
- [8] M. Englert, B. Vöcking, and M. Winkler. Economical caching with stochastic prices. In *Proceedings of the 5th Symposium on Stochastic Algorithms, Foundations and Applications (SAGA)*, pages 179–190, 2009.
- [9] A. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 385–394, 2005.
- [10] D. P. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1-2):7–35, 1999.
- [11] S. Göbbels. *Smart Caching for Continuous Broadband Services in Intermittent Wireless Networks*. PhD thesis, RWTH Aachen University, Chair of Communication Networks (ComNets), 2009.
- [12] J. Hannan. Approximation to bayes risk in repeated plays. *Contributions to the Theory of Games*, 3:97–139, 1957.
- [13] L. Johansson, M. Asbogard, and B. Egardt. Assessing the potential of predictive control for hybrid vehicle powertrains using stochastic dynamic programming. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):71–83, 2007.
- [14] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.
- [15] C.-Y. Li and G.-P. Liu. Optimal fuzzy power control and management of fuel cell/battery hybrid vehicles. *Journal of Power Sources*, 192(2):525–533, 2009.
- [16] N. Littlestone and M. Warmuth. The weighted majority algorithm. In *Proceedings of the 30th Symposium on Foundations of Computer Science (SFCS)*, pages 256–261, 1989.
- [17] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
- [18] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 928–936, 2003.

Learning rotations with little regret

Elad Hazan
IBM Almaden
650 Harry Road
San Jose, CA 95120
ehazan@cs.princeton.edu

Satyen Kale
Yahoo! Research
4301 Great America Parkway
Santa Clara, CA 95054
skale@yahoo-inc.com

Manfred K. Warmuth*
Department of Computer Science
UC Santa Cruz
manfred@cse.ucsc.edu

Abstract

We describe online algorithms for learning a rotation from pairs of unit vectors in \mathbb{R}^n . We show that the expected regret of our online algorithm compared to the best fixed rotation chosen offline is $O(\sqrt{nL})$, where L is the loss of the best rotation. We also give a lower bound that proves that this expected regret bound is optimal within a constant factor. This resolves an open problem posed in COLT 2008. Our online algorithm for choosing a rotation matrix in each trial is based on the Follow-The-Perturbed-Leader paradigm. It adds a random spectral perturbation to the matrix characterizing the loss incurred so far and then chooses the best rotation matrix for that loss. We also show that any deterministic algorithm for learning rotations has $\Omega(T)$ regret in the worst case.

1 Introduction

Rotations are a fundamental object in robotics and vision. The problem of learning rotations, or finding the underlying rotation from a given set of examples, has numerous applications in these areas (see [Aro09] for a summary of application areas). As a motivating example, in optical character recognition, rotational (or skew) correction is an important and challenging problem. Optically read characters need to be aligned before they can be recognized. A fast, low-regret online learning algorithm for rotations can be used for detecting skew using a small number of examples.

Besides their practical importance, rotations have been shown to be powerful enough to capture seemingly more general mappings. Rotations can represent arbitrary Euclidean transformations via a conformal embedding by adding two special dimensions [WCL05]. Also [DHSA93] showed the rotation group provides a universal representation for all Lie groups.

The batch learning problem has a simple and well known solution [Wah65, Sch66], but the question of whether there are online algorithms for this problem was posed in [SW08] as an open problem. Recently, an algorithm was given in [Aro09] based on the Matrix Exponentiated Gradient update [TRW05]. This algorithm elegantly exploits the Lie group/Lie algebra relationship between rotation matrices and skew symmetric matrices, respectively, and the matrix exponential and matrix logarithm that maps between these domains. However, this algorithm deterministically predicts with a single rotation matrix in each trial. In this paper, we prove that any such deterministic algorithm can be forced to have regret at least $\Omega(T)$, where T is the number of trials.

To achieve regret bounds that are sublinear in T , it is necessary to “hedge our bets” and predict randomly from a suitable distribution over rotation matrices. There are two ways in which this can be done: either the algorithm predicts deterministically with a parameter that represents a convex combination of rotation matrices or it explicitly produces a rotation matrix based on its internal randomization. Our algorithm is of the latter type and is in the Follow-The-Perturbed-Leader (FPL) [KV05] family of algorithms. At this point we do not know how to design algorithms of the former type. One promising approach makes use of the von Mises-Fisher distribution, which is an exponential family distribution over rotations (See discussion in [SW10]).

In this paper we bypass the differential geometry of rotations altogether and hedge by predicting with a random rotation matrix. The key insight is that the loss for our rotation learning problem is linear in the chosen rotation matrix. We add a suitably chosen random perturbation matrix to the matrix characterizing the loss incurred so far and then simply choose the best rotation matrix for the perturbed matrix. A good choice

*Supported by NSF grant IIS-0917397

of the perturbation matrix turns out to be a one that has exponentially distributed singular values and random orthogonal left and right singular vectors. Surprisingly, for this choice, we can show that the regret bound for the resulting algorithm is optimal within a constant factor.

Outline of paper: We begin with some preliminaries in the next section, the precise problem statement, and basic properties of rotations. In this section we also prove a lower bound for any deterministic algorithm. In section 3 we describe our randomized algorithm and give a bound on its expected regret. Following that, in section 4 we give the lower bound which applies to any algorithm, randomized or otherwise, and which matches the regret bound of our algorithm up to a constant factor.

2 Preliminaries and Problem Statement

2.1 Notation.

In this paper, all vectors lie in \mathbb{R}^n and all matrices in $\mathbb{R}^{n \times n}$. We use $\mathcal{SO}(n)$ to denote the special orthogonal group, i.e. the set of all rotation matrices \mathbf{R} . These are all orthogonal matrices of determinant one. For any vector \mathbf{x} , $\|\mathbf{x}\|$ denotes its ℓ_2 norm. For any matrix \mathbf{A} , $\|\mathbf{A}\|$ denotes its spectral norm (or Schatten- ∞ norm) which is the maximum singular value of \mathbf{A} . Furthermore, $\|\mathbf{A}\|_*$ denotes the trace norm (also known as the nuclear norm or Schatten-1 norm) which is the sum of all singular values. For two matrices \mathbf{A} and \mathbf{B} , $\mathbf{A} \bullet \mathbf{B}$ denotes the trace product $\text{Tr}(\mathbf{A}^\top \mathbf{B}) = \sum_{ij} A_{ij} B_{ij}$.

2.2 Online Learning of Rotations problem.

Learning proceeds in a series of trials. In every iteration for $t = 1, 2, \dots, T$:

1. The online learner is given a unit¹ vector \mathbf{x}_t (i.e. $\|\mathbf{x}_t\| = 1$).
2. The learner is then required to commit (either deterministically or probabilistically), to a rotation matrix $\mathbf{R}_t \in \mathcal{SO}(n)$. The choice of \mathbf{R}_t gives the predicted vector $\hat{\mathbf{y}}_t = \mathbf{R}_t \mathbf{x}_t$.
3. Finally the algorithm obtains true result, a unit vector \mathbf{y}_t (which is presumably the result of some unknown rotation applied to \mathbf{x}_t).
4. The loss to the learner then is half the squared norm of the difference between her predicted vector and the “true” rotated vector:

$$L_t(\mathbf{R}_t) = \frac{1}{2} \|\mathbf{R}_t \mathbf{x}_t - \mathbf{y}_t\|^2 = \frac{1}{2} [\|\mathbf{R}_t \mathbf{x}_t\|^2 + \|\mathbf{y}_t\|^2 - 2\mathbf{y}_t^\top \mathbf{R}_t \mathbf{x}_t] = 1 - (\mathbf{y}_t \mathbf{x}_t^\top) \bullet \mathbf{R}_t. \quad (1)$$

The last equality uses the fact that rotation preserves the ℓ_2 norm. Note that the loss is always in the range $[0, 2]$. The goal of the learner is to choose rotations \mathbf{R}_t in such a way as to minimize the regret on all T examples given by

$$\text{Regret}_T = \sum_{t=1}^T L_t(\mathbf{R}_t) - \min_{\mathbf{R} \in \mathcal{SO}(n)} \sum_{t=1}^T L_t(\mathbf{R}).$$

We aim to find online algorithms with regret sublinear in T (Such algorithms are called Hannan consistent [CBL06]). Henceforth we give an algorithm whose regret is optimal within a constant factor.

We can also consider a setting of the problem that is more difficult for the learner in which \mathbf{x}_t is not known at the point when the rotation matrix must be chosen. In this setting the learner first commits to a distribution over rotations and only then sees a pair $\mathbf{x}_t, \mathbf{y}_t$ and incurs the associated expected loss. Our algorithm works even in this setting and the bound on its expected regret remains valid. However our lower bounds apply to the above definition that is easier for the learner.

2.3 Main Result.

Our main result is a Follow-The-Perturbed-Leader type algorithm (see Section 3) with the following guarantee on its expected regret:

Theorem 1 *There is a randomized online algorithm which for any sequence of T examples for which the loss L of the best fixed rotation in hindsight, i.e. $L = \min_{\mathbf{R} \in \mathcal{SO}(n)} \sum_{t=1}^T L_t(\mathbf{R})$, is at least $16n$, achieves regret*

$$\mathbf{E}[\text{Regret}_T] \leq O(\sqrt{nL}).$$

This algorithm can be implemented to run in $O(n^3)$ time per trial.

¹Note that there is no loss of generality in restricting \mathbf{x}_t and \mathbf{y}_t to be unit vectors. Our algorithm and its analysis, work unchanged assuming only $\|\mathbf{y}_t \mathbf{x}_t^\top\|_* = \|\mathbf{x}_t\| \|\mathbf{y}_t\| \leq 1$.

We also can prove a matching lower bound.

Theorem 2 For any integer $T > n$, and for any online algorithm for learning rotations, there is a sequence of T examples, such that the algorithm incurs regret at least $\Omega(\sqrt{nT})$.

Since the per trial loss of any rotation is at most 2, the loss L of the best rotation chosen in hindsight is at most $2T$, and therefore any lower bound of $\Omega(\sqrt{nT})$ on the regret implies a lower bound of $\Omega(\sqrt{nL})$. This shows that the regret of our algorithm (given in Theorem 1 above) is tight up to constant factors.

2.4 Solving the Offline Problem.

Before describing our online algorithm, we need to understand how to solve the optimization problem of offline (batch) algorithm:

$$\operatorname{argmin}_{\mathbf{R} \in \mathcal{SO}(n)} \sum_{t=1}^T L_t(\mathbf{R}) = \operatorname{argmax}_{\mathbf{R} \in \mathcal{SO}(n)} \sum_{t=1}^T (\mathbf{y}_t \mathbf{x}_t^\top) \bullet \mathbf{R}.$$

The equality follows from rewriting the loss function L_t as in (1). In general, an optimization problem of the form

$$\operatorname{argmax}_{\mathbf{R} \in \mathcal{SO}(n)} \mathbf{M} \bullet \mathbf{R}$$

for some matrix \mathbf{M} , is a classical problem known as Wahba's problem [Wah65]. Figure 1 gives a simple example of the challenges in solving Wahba's problem: degeneracies can arise unexpectedly.

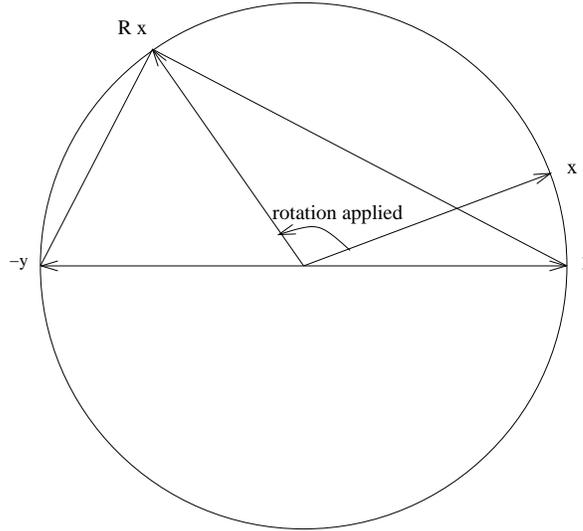


Figure 1: If we have two examples (\mathbf{x}, \mathbf{y}) and $(\mathbf{x}, -\mathbf{y})$, then regardless of which rotation \mathbf{R} we choose, $\mathbf{R}\mathbf{x}$ has loss exactly 2. This is a consequence of the geometric fact that the diameter connecting \mathbf{y} to $-\mathbf{y}$ subtends a right angle at $\mathbf{R}\mathbf{x}$, and therefore Pythagoras' theorem applies. Algebraically, for any \mathbf{R} , we have $L_1(\mathbf{R}) + L_2(\mathbf{R}) = 2 - (\mathbf{y}\mathbf{x}^\top - \mathbf{y}\mathbf{x}^\top) \bullet \mathbf{R} = 2 - 0 = 2$.

Nevertheless, Wahba's problem has a very elegant solution² via any singular value decomposition (SVD) of \mathbf{M} .

Lemma 3 Let $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be any SVD of \mathbf{M} , i.e. \mathbf{U} and \mathbf{V} are orthogonal matrices, and $\mathbf{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$ is the diagonal matrix of non-negative singular values. Assume that σ_n is the smallest singular value. Let $s := \det(\mathbf{U}) \det(\mathbf{V})$, where \det denotes determinant. Since \mathbf{U} and \mathbf{V} are orthogonal, $s \in \{+1, -1\}$. Now if $\mathbf{W} := \operatorname{diag}(1, 1, \dots, 1, s)$, then $\mathbf{U}\mathbf{W}\mathbf{V}^\top$ is a solution to Wahba's problem, i.e.

$$\mathbf{U}\mathbf{W}\mathbf{V}^\top \in \operatorname{argmax}_{\mathbf{R} \in \mathcal{SO}(n)} \mathbf{M} \bullet \mathbf{R},$$

and the value of the optimal solutions is $\sum_{i=1}^{n-1} \sigma_i + s\sigma_n$, which is always non-negative.

²Note that the solution to Wahba's problem may not be unique, in which case we are satisfied with any solution amongst the optimal set of solutions.

This solution is a rotation matrix since it is a product of three orthogonal matrices, and its determinant equals $\det(\mathbf{U}) \det(\mathbf{V}) \det(\mathbf{W}) = 1$. The solution can be found in $O(n^3)$ time by constructing a SVD of \mathbf{M} .

We have been unable to find a complete proof of this lemma for dimensions more than 3 in the literature and for the sake of completeness we give a self-contained proof in Appendix A.

Note that if we are simply optimizing over all orthogonal matrices $\mathbf{R} \in \mathcal{O}(n)$, with no condition on $\det(\mathbf{R})$, then the construction becomes simpler (also proven in Appendix A):

Lemma 4 *Let $\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^\top$ be a SVD of \mathbf{M} as in Lemma 3. Then*

$$\mathbf{UV}^\top \in \operatorname{argmax}_{\mathbf{R} \in \mathcal{O}(n)} \mathbf{M} \bullet \mathbf{R}$$

and the value of the optimum solutions is $\sum_{i=1}^n \sigma_i$.

This is also a classical problem known as the ‘‘Orthogonal Procrustes Problem’’, first solved by Schönemann [Sch66].

2.5 The necessity of randomization in learning rotations.

We give an adversary strategy that forces any deterministic algorithm such as the simple Follow-The-Leader algorithm (which predicts with the rotation matrix minimizing the loss on the past examples) or Arora’s deterministic algorithm [Aro09] to have linear regret. This shows that randomization is essential for obtaining good regret bounds for learning rotations.

Theorem 5 *Any algorithm which deterministically predicts with a single rotation matrix at each trial has worst case regret at least T on example sequences of length T .*

Proof: Consider the following problem instance. In each iteration, the adversary always sets $\mathbf{x}_t = \mathbf{e}_1$. Since the algorithm is deterministic, the adversary can then compute the matrix \mathbf{R}_t in every iteration. The algorithm predicts with $\hat{\mathbf{y}}_t = \mathbf{R}_t \mathbf{x}_t$ and the adversary chooses \mathbf{y}_t as $-\hat{\mathbf{y}}_t$. Therefore the algorithm’s per trial loss is

$$\frac{1}{2} \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|^2 = \frac{1}{2} \|2\hat{\mathbf{y}}_t\|^2 = 2,$$

amounting to a loss $2T$ in all trials.

On the other hand the loss of the optimum rotation is

$$\min_{\mathbf{R} \in \mathcal{SO}(n)} \frac{1}{2} \|\mathbf{R} \mathbf{x}_t - \mathbf{y}_t\|^2 = T - \max_{\mathbf{R} \in \mathcal{SO}(n)} \mathbf{W}_T \bullet \mathbf{R},$$

where $\mathbf{W}_T = \sum_{t=1}^T \mathbf{y}_t \mathbf{x}_t^\top$. By Lemma 3, $\max_{\mathbf{R} \in \mathcal{SO}(n)} \mathbf{W}_T \bullet \mathbf{R} \geq 0$. Thus, the loss of the optimum rotation is at most T . Hence, the algorithm has regret at least $2T - T = T$. ■

In view of this lower bound, only randomized algorithms can hope to achieve sublinear regret. As we observed in equation (1), the square loss over the (non-convex) set $\mathcal{SO}(n)$ is linear. Therefore it is natural to apply the Follow-The-Perturbed-Leader (FPL) type algorithm [KV05], as this generic template is capable of handling non-convex decision sets. A direct application of the Kalai-Vempala result gives the following algorithm:

$$\mathbf{R}_t = \operatorname{argmin}_{\mathbf{R} \in \mathcal{SO}(n)} \sum_{i=1}^{t-1} L_i(\mathbf{R}) - \mathbf{N} \bullet \mathbf{R},$$

where \mathbf{N} is a random matrix whose entries are i.i.d. uniform random numbers in the range $[0, \frac{1}{\varepsilon}]$ for some parameter ε . However, tuning ε to its optimal value gives a suboptimal regret bound of $O(n^{5/4} \sqrt{T})$. The reason for this suboptimality is that uniform sampling of the components of \mathbf{N} does not match the characteristics of our problem.

2.6 Sampling Random Orthogonal Matrices.

A fundamental task in our better implementation of FPL is sampling of random orthogonal matrices ‘‘uniformly’’, in the sense that the density of the distribution at any two matrices \mathbf{U} and \mathbf{V} in $\mathcal{O}(n)$ is the same. Technically, this distribution is given by the Haar measure ν on $\mathcal{O}(n)$ scaled so that $\nu(\mathcal{O}(n)) = 1$. The distribution ν has the property that for any fixed orthogonal matrix $\mathbf{U} \in \mathcal{O}(n)$, if \mathbf{V} is sampled from ν , then the distribution on $\mathcal{O}(n)$ induced by \mathbf{UV} is also ν . This implies the desired uniformity property.

To sample a random orthogonal matrix from such a uniform distribution is essentially to choose a random orthogonal basis. The following process generates such basis incrementally: start with a random unit vector, and then repeatedly pick a random unit vector orthogonal to all vectors seen so far. This process can be implemented by running the Gram-Schmidt process on a set of n random vectors. Indeed, the efficient way of doing essentially this is by using the QR-decomposition of a random matrix with independent standard Gaussian entries (see [Ste80] for more details).

3 Algorithm and Analysis

Instead of using uniform randomness, the correct perturbation matrix turns out to be one with singular values chosen from an exponential distribution with randomly chosen left and right singular vectors, as given in the algorithm below. The online algorithm can be implemented in $O(n^3)$ time per trial by uniformly sampling orthogonal matrices as outlined in Section 2.6 and by optimizing a linear function over rotations as done in the solution of the off-line problem (see Sections 2.4).

Algorithm 1 FSPL: Follow-The-Spectrally-Perturbed-Leader

- 1: Select n non-negative real numbers $\sigma_1, \sigma_2, \dots, \sigma_n$ independently from the exponential distribution with rate parameter ε , i.e. with density $\varepsilon \exp(-\varepsilon\sigma) d\sigma$. Let $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$.
- 2: Select two random orthogonal matrices $\mathbf{U}, \mathbf{V} \in \mathcal{O}(n)$ uniformly from the Haar measure.
- 3: Define $\mathbf{N} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$.
- 4: **for** $t = 1$ to T **do**
- 5: Let \mathbf{R}_t be the result of the following optimization problem:

$$\mathbf{R}_t = \underset{\mathbf{R} \in \mathcal{SO}(n)}{\text{argmin}} \sum_{i=1}^{t-1} L_i(\mathbf{R}) - \mathbf{N} \bullet \mathbf{R} = \underset{\mathbf{R} \in \mathcal{SO}(n)}{\text{argmax}} \left[\sum_{i=1}^{t-1} (\mathbf{y}_i \mathbf{x}_i^\top) + \mathbf{N} \right] \bullet \mathbf{R}.$$

- 6: Obtain vector \mathbf{x}_t . Predict $\hat{\mathbf{y}}_t = \mathbf{R}_t \mathbf{x}_t$ and observe the result vector \mathbf{y}_t . Suffer loss $L_t(\mathbf{R}_t)$.
 - 7: **end for**
-

We point out that our algorithm is *basis-invariant* in the following sense: fix an orthogonal matrix (i.e. an orthonormal basis) $\mathbf{B} \in \mathcal{O}(n)$. Consider two problem instances, one with examples $(\mathbf{x}_t, \mathbf{y}_t)$ for $t = 1, 2, \dots, T$, and another with the same examples expressed in the alternate basis \mathbf{B} , viz $(\mathbf{B}\mathbf{x}_t, \mathbf{B}\mathbf{y}_t)$, and consider running the algorithm on these two sequences of examples. Geometrically, the instances are the same, so it is desirable for the algorithm to behave the same way, modulo the change of basis.

In our algorithm, the orthogonal matrices \mathbf{U} and \mathbf{V} are drawn from the uniform Haar measure over orthogonal matrices, the distribution of \mathbf{N} and that of $\mathbf{B}\mathbf{N}\mathbf{B}^\top$ is the same. This fact allows us to correlate the choice of the perturbations in the algorithms running over the two instances by choosing the perturbation \mathbf{N} for the first instance, and the perturbation $\mathbf{B}\mathbf{N}\mathbf{B}^\top$ for the second instance.

Thus, if

$$\mathbf{R}_t = \underset{\mathbf{R} \in \mathcal{SO}(n)}{\text{argmax}} \left[\sum_{i=1}^{t-1} (\mathbf{y}_i \mathbf{x}_i^\top) + \mathbf{N} \right] \bullet \mathbf{R},$$

then

$$\mathbf{B}\mathbf{R}_t\mathbf{B}^\top = \underset{\mathbf{R} \in \mathcal{SO}(n)}{\text{argmax}} \left[\sum_{i=1}^{t-1} (\mathbf{B}\mathbf{y}_i (\mathbf{B}\mathbf{x}_i)^\top) + \mathbf{B}\mathbf{N}\mathbf{B}^\top \right] \bullet \mathbf{R}.$$

Note that $\mathbf{B}\mathbf{R}_t\mathbf{B}^\top$ is simply the rotation matrix \mathbf{R}_t expressed in basis \mathbf{B} . The prediction of $\mathbf{B}\mathbf{R}_t\mathbf{B}^\top$ on the transformed example $(\mathbf{B}\mathbf{x}_t, \mathbf{B}\mathbf{y}_t)$ is $\mathbf{B}\mathbf{R}_t\mathbf{B}^\top \mathbf{B}\mathbf{x}_t$ which simplifies to $\mathbf{R}_t \mathbf{x}_t$. This prediction is the same as the prediction of \mathbf{R}_t on the original example $(\mathbf{x}_t, \mathbf{y}_t)$, i.e. $\mathbf{R}_t \mathbf{x}_t$, and while premultiplying with the transformation \mathbf{B} . Geometrically, our algorithms is doing the same thing on the original and the transformed sequence of examples, and the losses are the same well.

Before launching into the analysis, we make a few observations regarding the noise matrix \mathbf{N} . Essentially, the noise matrix is sampled by constructing each component of its SVD randomly. With probability 1, the singular values of \mathbf{N} , viz. $\sigma_1, \sigma_2, \dots, \sigma_n$ are all distinct, so in the sequel we assume this is the case whenever we talk about the noise matrix \mathbf{N} .

In the induced probability distribution over matrices \mathbf{N} , if the SVD of \mathbf{N} is $\mathbf{N} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, the density of the distribution at \mathbf{N} is

$$d\mu(\mathbf{N}) = 2^n n! \varepsilon^n \exp \left[-\varepsilon \sum_{i=1}^n \sigma_i \right] d\sigma_1 d\sigma_2 \dots d\sigma_n d\nu(\mathbf{U}) d\nu(\mathbf{V}),$$

where $d\nu(\mathbf{U})$ is the density at \mathbf{U} in the Haar measure over $\mathcal{O}(n)$. The leading factor of $2^n n!$ is because the SVD is uniquely determined by the ordering of the singular values and the sign multiplying the n pairs of right and left singular vectors. Note that $\sum_{i=1}^n \sigma_i = \|\mathbf{N}\|_*$, the trace norm of \mathbf{N} . Thus, since ν is the uniform Haar measure on $\mathcal{O}(n)$, we may (informally) say that

$$d\mu(\mathbf{N}) \propto \exp(-\varepsilon \|\mathbf{N}\|_*).$$

Now, we can analyze FSPL. The following Theorem implies Theorem 1.

Theorem 6 For any target rotation matrix \mathbf{R}^* , the algorithm FSPL attains the following regret bound:

$$\mathbf{E} \left[\sum_{t=1}^T L_t(\mathbf{R}_t) \right] - \sum_{t=1}^T L_t(\mathbf{R}^*) \leq \frac{\varepsilon}{1-\varepsilon} \sum_{t=1}^T L_t(\mathbf{R}^*) + \frac{2n}{(1-\varepsilon)\varepsilon}.$$

If $L \geq 16n$ and ε is set to $\sqrt{\frac{n}{L}}$, where $L = \sum_{t=1}^T L_t(\mathbf{R}^*)$, then the expected regret is bounded³ by $4\sqrt{nL}$.

Proof: The analysis of FSPL is based on the technique of Kalai and Vempala [KV05] for analyzing the Follow-The-Perturbed-Leader style algorithms.

For convenience of notation, define $\mathbf{W}_t = \sum_{i=1}^{t-1} \mathbf{y}_i \mathbf{x}_i^\top$, and the function $\mathcal{R} : \mathbb{R}^{n \times n} \rightarrow \mathcal{SO}(n)$ to be

$$\mathcal{R}(\mathbf{M}) := \operatorname{argmax}_{\mathbf{R} \in \mathcal{SO}(n)} \mathbf{M} \bullet \mathbf{R}.$$

Now the rotation matrix \mathbf{R}_t chosen by the algorithm is denoted as $\mathcal{R}(\mathbf{W}_t + \mathbf{N})$. The first step of analyzing any Follow-The-Perturbed-Leader algorithm, is the following inequality which is essentially proved in [KV05], but we give the proof in Appendix B for the sake of completeness.

Claim 7 For any target rotation matrix \mathbf{R}^* , we have

$$\sum_{t=1}^T L_t(\mathbf{R}_t) - \sum_{t=1}^T L_t(\mathbf{R}^*) \leq \sum_{t=1}^T L_t(\mathbf{R}_t) - L_t(\mathbf{R}_{t+1}) + \mathbf{N} \bullet \mathbf{R}_1 - \mathbf{N} \bullet \mathbf{R}^*.$$

Now, to bound the *expected* regret, we prove the following two bounds:

$$\mathbf{E}[L_t(\mathbf{R}_t) - L_t(\mathbf{R}_{t+1})] \leq \varepsilon \mathbf{E}[L_t(\mathbf{R}_t)], \quad (2)$$

and

$$\mathbf{E}[\mathbf{N} \bullet (\mathbf{R}_1 - \mathbf{R}^*)] \leq \frac{2n}{\varepsilon}. \quad (3)$$

Combining these two bounds, and some algebra, we get the stated bound on the regret:

$$\mathbf{E} \left[\sum_{t=1}^T L_t(\mathbf{R}_t) \right] - \sum_{t=1}^T L_t(\mathbf{R}^*) \leq \frac{\varepsilon}{1-\varepsilon} \sum_{t=1}^T L_t(\mathbf{R}^*) + \frac{2n}{(1-\varepsilon)\varepsilon}.$$

If we set $\varepsilon = \sqrt{\frac{n}{L}}$, where $L = \sum_{t=1}^T L_t(\mathbf{R}^*)$, then the expected regret is bounded by $\mathbf{E}[\text{Regret}] \leq 4\sqrt{nL}$, assuming⁴ $L \geq 16n$.

We prove the inequality (2) now. Since we are only interested in expected values, it doesn't matter whether we choose the noise at the beginning, or if we re-randomize every trial. Thus, denoting by \mathbf{N} and \mathbf{N}' the noise chosen in the t^{th} and $(t+1)^{\text{st}}$ trials respectively, we have

$$\begin{aligned} & \mathbf{E}[L_t(\mathbf{R}_t)] - \mathbf{E}[L_t(\mathbf{R}_{t+1})] \\ &= \int_{\mathbf{N}} L_t(\mathcal{R}(\mathbf{W}_t + \mathbf{N})) d\mu(\mathbf{N}) - \int_{\mathbf{N}'} L_t(\mathcal{R}(\mathbf{W}_{t+1} + \mathbf{N}')) d\mu(\mathbf{N}') \\ &= \int_{\mathbf{N}} L_t(\mathcal{R}(\mathbf{W}_t + \mathbf{N})) d\mu(\mathbf{N}) - L_t(\mathcal{R}(\mathbf{W}_t + \mathbf{N})) d\mu(\mathbf{N} - \mathbf{y}_t \mathbf{x}_t^\top) \\ & \text{(doing a change of variables in the integration: } \mathbf{N}' = \mathbf{N} - \mathbf{y}_t \mathbf{x}_t^\top) \\ &\leq \int_{\mathbf{N}} \varepsilon L_t(\mathcal{R}(\mathbf{W}_t + \mathbf{N})) d\mu(\mathbf{N}) \\ &= \varepsilon \mathbf{E}[L_t(\mathbf{R}_t)]. \end{aligned}$$

The last inequality follows from the fact that for any \mathbf{N} , we have

$$d\mu(\mathbf{N}) - d\mu(\mathbf{N} - \mathbf{y}_t \mathbf{x}_t^\top) \leq \varepsilon d\mu(\mathbf{N}),$$

which we prove now:

$$\frac{d\mu(\mathbf{N} - \mathbf{y}_t \mathbf{x}_t^\top)}{d\mu(\mathbf{N})} = \frac{\exp(-\varepsilon \|\mathbf{N} - \mathbf{y}_t \mathbf{x}_t^\top\|_\star)}{\exp(-\varepsilon \|\mathbf{N}\|_\star)} \geq \exp(-\varepsilon \|\mathbf{y}_t \mathbf{x}_t^\top\|_\star) \geq \exp(-\varepsilon) \geq 1 - \varepsilon.$$

³Standard techniques such as the ‘‘doubling trick’’ (see e.g. [CBFH⁺97]), can be used to obtain the same type of regret bound even if L is not known in advance while only slightly increasing the constant in front of the square root.

⁴A slightly better constant is obtainable via some further optimization over ε .

We use the triangle inequality for the $\|\cdot\|_*$ norm in the first inequality, and $\|\mathbf{y}_t \mathbf{x}_t^\top\|_* \leq 1$ in the second inequality.

Now, we prove inequality (3). To this end, we show that for any rotation matrix \mathbf{R} , we have $\mathbf{E}[\|\mathbf{N} \bullet \mathbf{R}\|] \leq \frac{n}{\varepsilon}$. To prove this, note that

$$\|\mathbf{N} \bullet \mathbf{R}\| \leq \|\mathbf{N}\|_* \|\mathbf{R}\|,$$

by the Hölder inequality for trace norm (Schatten 1-norm) and its dual matrix norm (Schatten ∞ -norm). Clearly, $\|\mathbf{R}\| = 1$ since all singular values of an orthogonal matrix are one. For the noise matrix \mathbf{N} , if $\sigma_1, \sigma_2, \dots, \sigma_n$ are its singular values, then

$$\mathbf{E}[\|\mathbf{N}\|_*] = \mathbf{E}\left[\sum_{i=1}^n \sigma_i\right] = \sum_{i=1}^n \mathbf{E}[\sigma_i] = \frac{n}{\varepsilon}$$

since the σ_i 's are independently distributed exponential random variables with mean $\frac{1}{\varepsilon}$. Putting the bounds together, we get that

$$\mathbf{E}[\mathbf{N} \bullet (\mathbf{R}_1 - \mathbf{R}^*)] \leq \frac{2n}{\varepsilon}. \quad \blacksquare$$

4 Lower Bound

We now show a lower bound against any algorithm (including probabilistic ones). This matches the upper bound for the FSPL algorithm up to constant factors.

Proof of Theorem 2. We assume for convenience that the dimension is $n + 1$, rather than n . For any online algorithm for the rotations problem we construct an example sequence of length T for which this algorithm has regret $\Omega(\sqrt{nT})$.

Let \mathbf{e}_i denote the i -th standard basis vector, i.e. the vector with 1 in its i -th coordinate and 0 everywhere else. In trial $t < T$, set $\mathbf{x}_t = \mathbf{e}_{f(t)}$, where $f(t) = (t \bmod n) + 1$ (i.e., cycle through the coordinates $1, 2, \dots, n$), and $\mathbf{y}_t = \sigma_t \mathbf{e}_{f(t)}$, where $\sigma_t \in \{-1, 1\}$ uniformly at random. For any coordinate $i \in 1, 2, \dots, n$, let $X_i = \sum_{t: f(t)=i} \sigma_t$. For the final trial T , set $\mathbf{x}_T = \mathbf{e}_{n+1}$, and $\mathbf{y}_T = \sigma_T \mathbf{e}_{n+1}$, where $\sigma_T \in \{-1, 1\}$ is chosen in a certain way specified momentarily. First, note that

$$\mathbf{W}_{T+1} = \sum_{t=1}^T \mathbf{y}_t \mathbf{x}_t^\top = \text{diag}(X_1, X_2, \dots, X_n, \sigma_T).$$

We choose σ_T so that

$$\det(\mathbf{W}_{T+1}) = \sigma_T \prod_{i=1}^n X_i > 0.$$

In other words, $\sigma_T = \text{sgn}(\prod_{i=1}^n X_i)$.

By Lemma 3, the solution to the offline problem is the rotation matrix $\mathbf{R}^* = \arg\max_{\mathbf{R} \in \mathcal{SO}(n)} \mathbf{W}_{T+1} \bullet \mathbf{R}$, where

$$\mathbf{R}^* = \text{diag}(\text{sgn}(X_1), \text{sgn}(X_2), \dots, \text{sgn}(X_n), \sigma_T),$$

and the loss of this matrix is

$$\sum_{t=1}^T L_t(\mathbf{R}^*) = T - \mathbf{W}_{T+1} \bullet \mathbf{R}^* = T - \sum_{i=1}^n |X_i| - 1.$$

Since each X_i is a sum of $\lfloor \frac{T-1}{n} \rfloor$ Rademacher variables (give or take 1), standard probabilistic bounds (such as Khintchine's inequality [Haa82]) imply that $\mathbf{E}[|X_i|] = \Omega(\sqrt{T/n})$, where the expectation is taken over the choice of the σ_t 's. Thus, the expected loss of the optimal rotation is bounded as follows:

$$\mathbf{E}\left[\sum_{t=1}^T L_t(\mathbf{R}^*)\right] = T - \mathbf{E}[\mathbf{W}_{T+1} \bullet \mathbf{R}^*] = T - \Omega(\sqrt{T/n} \cdot n) = T - \Omega(\sqrt{nT}).$$

Finally, note that for $t < T$, regardless of which specific rotation matrix \mathbf{R}_t is selected by the algorithm,

$$\mathbf{E}_{\sigma_t}[L_t(\mathbf{R}_t)] = 1 - \mathbf{E}_{\sigma_t}[\mathbf{y}_t \mathbf{x}_t^\top \bullet \mathbf{R}_t] = 1 - \mathbf{E}_{\sigma_t}[\sigma_t] \cdot \mathbf{e}_{f(t)}^\top \mathbf{R}_t \mathbf{e}_{f(t)} = 1.$$

In trial T , the algorithm might at best have a loss of 0. Thus, the expected loss of the algorithm is at least $T - 1$, and hence its expected regret is $\Omega(\sqrt{nT})$. This implies that there is a choice of the σ_t 's so that the actual regret of the algorithm is $\Omega(\sqrt{nT})$, as required. \blacksquare

5 Conclusions

We have presented tight bounds on the regret for online learning of rotation matrices. Our main technique is a Follow-The-Perturbed-Leader type algorithm with spectral perturbations. Essentially the same algorithm works in a different setting as well: online learning of a basis (or more colorfully, the Online Orthogonal Procrustes problem). Here, the learner is presented the example \mathbf{x}_t , and now the learner chooses an orthogonal matrix $\mathbf{U}_t \in \mathcal{O}(n)$ instead of a rotation matrix and predicts $\hat{\mathbf{y}}_t = \mathbf{U}_t \mathbf{x}_t$. This corresponds to a change of basis. Then the actual vector \mathbf{y}_t is revealed, and the loss is defined the same way: $\frac{1}{2} \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|^2$. The goal is to minimize regret with respect to the best change of basis in hindsight. It is apparent from the algorithm's analysis that we can use the same algorithm with the same perturbations, except that we optimize over $\mathcal{O}(n)$ rather than $\mathcal{SO}(n)$, an even simpler task (c.f. Lemma 4). Our regret bounds carry over to this setting.

It would be very interesting to find other applications of the spectral perturbations idea. In particular, the matrix version of the FPL algorithm might lead to speedups of the Matrix Hedge [WK06] and more generally the Matrix Exponentiated Gradient algorithm which both optimize over density matrices. To realize these speedups one would either have to get away with a single perturbation matrix or perturbation matrices that can be sampled in $O(n^2)$ time per trial. We expand on these ideas in an open problem posed in this conference [HKW10].

Acknowledgement

Our work was motivated by preliminary research done with Adam Smith [SW10, SW08] and we greatly benefited from discussions with him. We are also thankful to Abhishek Kumar for allowing us to include a simplified proof of Wahba's problem which was worked out in collaboration with him.

References

- [Aro09] R. Arora. On learning rotations. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 55–63. MIT Press, 2009.
- [CBFH⁺97] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.
- [CBL06] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.
- [DHSA93] C. Doran, D. Hestenes, F. Sommen, and N. Van Acker. Lie groups as spin groups. *J. Math. Phys.*, 34(8):3642–3669, August 1993.
- [Haa82] U. Haagerup. The best constants in the Khintchine inequality. *Studia Math.*, 70(3):427–485, 1982.
- [HKW10] E. Hazan, S. Kale, and M. K. Warmuth. On-line variance minimization in $O(n^2)$ per trial? In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT '10)*, 2010.
- [KV05] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005.
- [Sch66] P. Schonemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1), March 1966.
- [Ste80] G. W. Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM J. Numer. Anal.*, 17(3):403–409, 1980.
- [SW08] A. Smith and M. K. Warmuth. Learning rotations. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT '08)*, page 517, July 2008.
- [SW10] A. M. Smith and M. K. Warmuth. Learning rotations online. Technical Report UCSC-SOE-10-08, Department of Computer Science, University of California, Santa Cruz, February 2010.
- [TRW05] K. Tsuda, G. Rätsch, and M. K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projections. *Journal of Machine Learning Research*, 6:995–1018, June 2005.
- [Wah65] G. Wahba. Problem 65-1, a least squares estimate of satellite attitude. *SIAM Review*, 7(3), July 1965.

[WCL05] R. Wareham, J. Cameron, and J. Lasenby. Applications of conformal geometric algebra in computer vision and graphics. *6th International Workshop IWMM 2004*, pages 329–349, 2005.

[WK06] M. K. Warmuth and D. Kuzmin. Online variance minimization. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT '06)*, 2006.

A Solutions of Wahba’s problem and the Orthogonal Procrustes Problem

We first prove Lemma 4, since it is simpler and it gives a reduction which will be useful for the proof of Lemma 3.

Proof of Lemma 4: Recall that we want to compute

$$\max_{\mathbf{R} \in \mathcal{O}(n)} \mathbf{M} \bullet \mathbf{R}.$$

Let $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be an SVD of \mathbf{M} , so that \mathbf{U} and \mathbf{V} are orthogonal matrices, and $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$ is a diagonal matrix of the non-negative singular values σ_i .

Now, we do a change of variables. Instead of maximizing over an orthogonal matrix \mathbf{R} , we maximize over orthogonal matrix $\mathbf{W} = \mathbf{U}^\top \mathbf{R} \mathbf{V}$. This lets us rewrite the dot product we are minimizing over

$$\mathbf{M} \bullet \mathbf{R} = \text{Tr}(\mathbf{M}^\top \mathbf{U} \mathbf{W} \mathbf{V}^\top) = \text{Tr}(\mathbf{V} \mathbf{\Sigma}^\top \mathbf{U}^\top \mathbf{U} \mathbf{W} \mathbf{V}^\top) = \text{Tr}(\mathbf{\Sigma}^\top \mathbf{W}) = \mathbf{\Sigma} \bullet \mathbf{W}. \quad (4)$$

Since \mathbf{W} is an orthogonal matrix, we have $|W_{ii}| \leq 1$ for all i . Hence, the linear expression $\mathbf{\Sigma} \bullet \mathbf{W} = \sum_i \sigma_i W_{ii}$ is maximized when $\mathbf{W} = \mathbf{I}$, the identity matrix. We conclude that $\mathbf{R} = \mathbf{U} \mathbf{V}^\top$ is an optimal solution to $\max_{\mathbf{R} \in \mathcal{O}(n)} \mathbf{M} \bullet \mathbf{R}$ and all maxima has the value $\sum_{i=1}^n \sigma_i$. ■

We have been unable to find a complete, rigorous solution of Wahba’s problem in the literature for dimensions more than 3. For the sake of completeness, we give a complete proof. This proof was obtained in collaboration with Abhishek Kumar, simplifying a previous version given in the submitted version of this paper.

Proof of Lemma 3: Recall that we want to compute

$$\max_{\mathbf{R} \in \mathcal{SO}(n)} \mathbf{M} \bullet \mathbf{R}.$$

As in the proof of Lemma 4, let $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be an SVD of \mathbf{M} . As before, we do a change of variables from a rotation matrix \mathbf{R} to an orthogonal matrix $\mathbf{W} = \mathbf{U}^\top \mathbf{R} \mathbf{V}$, with the following condition on the determinant of \mathbf{W} :

$$\det(\mathbf{W}) = \det(\mathbf{U}) \det(\mathbf{R}) \det(\mathbf{V}) = \det(\mathbf{U}) \det(\mathbf{V}) =: s \in \{+1, -1\}.$$

Using equation (4), the problem now reduces to:

$$\max_{\mathbf{W} \in \mathcal{O}(n), \det(\mathbf{W})=s} \mathbf{\Sigma} \bullet \mathbf{W}. \quad (5)$$

The case $\det(\mathbf{W}) = 1$ is easy. We already showed in the previous lemma that

$$\mathbf{I} \in \operatorname{argmax}_{\mathbf{W} \in \mathcal{O}(n)} \mathbf{\Sigma} \bullet \mathbf{W}.$$

Thus in this case the constraint on the determinant of \mathbf{W} is immaterial and the value of the maxima is $\sum_{i=1}^n \sigma_i = \sum_{i=1}^{n-1} \sigma_i + s\sigma_n$, where σ_n is the smallest singular value.

The case $\det(\mathbf{W}) = -1$ is considerably harder. We need to show $\mathbf{W} = \text{diag}(1, 1, \dots, 1, -1)$ is an optimal solution which has value $\sum_{i=1}^{n-1} \sigma_i - \sigma_n = \sum_{i=1}^{n-1} \sigma_i + s\sigma_n$, where σ_n is the smallest singular value.

Let \mathbf{W} be an arbitrary orthogonal matrix of determinant -1 . We make the following observations regarding \mathbf{W} . First, if $\lambda_1, \lambda_2, \dots, \lambda_n$ are the n (real or complex) eigenvalues of \mathbf{W} , then we have

$$\prod_{i=1}^n \lambda_i = \det(\mathbf{W}) = -1.$$

Since $\mathbf{W} \in \mathcal{O}(n)$, all eigenvalues λ_i have magnitude $|\lambda_i| = 1$: this is because if λ is an eigenvalue of \mathbf{W} with eigenvector \mathbf{v} , i.e. $\mathbf{W}\mathbf{v} = \lambda\mathbf{v}$, then

$$1 = \|\mathbf{v}\| = \|\mathbf{W}\mathbf{v}\| = \|\lambda\mathbf{v}\| = |\lambda| \|\mathbf{v}\| = |\lambda|,$$

where the second equality uses $\mathbf{W} \in \mathcal{O}(n)$.

We claim that at least one eigenvalue of the matrix \mathbf{W} is -1 . This is so because all the complex eigenvalues of the *real* matrix \mathbf{M} must occur in complex conjugate pairs, $a + ib$ and $a - ib$, for some $b \neq 0$ and $a^2 + b^2 = 1$. Now, the product of any such complex conjugate pair of eigenvalues is $(a + ib)(a - ib) = a^2 + b^2 = 1$. Hence, the product of all complex eigenvalues is 1. Since the product of all eigenvalues (real or complex) is -1 , and all real eigenvalues are either $+1$ or -1 , we must have at least one eigenvalue being -1 .

For convenience of notation, let $\lambda_n = -1$. Now, we have

$$\text{tr}(\mathbf{W}) = \sum_{i=1}^{n-1} \lambda_i + \lambda_n = \sum_{i=1}^{n-1} \text{real}(\lambda_i) - 1 \leq n - 1 - 1 = n - 2.$$

Here, $\text{real}(z)$ is the real part of a complex number z , and we use the fact that the sum of two complex conjugate eigenvalues $a + ib$ and $a - ib$ is $2a$, which is the sum of their real parts. We also used the fact that for any eigenvalue λ_i , $\text{real}(\lambda_i) \leq 1$ since $|\lambda_i| = 1$.

Finally, note that $|W_{ii}| \leq 1$ since \mathbf{W} is an orthogonal matrix. Now, consider the following linear program which is a relaxation of the optimization problem (5) (This is a relaxation since the last inequality holds for all solutions of (5) and we drop the constraint that \mathbf{W} is an orthogonal matrix of determinant -1):

$$\begin{aligned} & \max \sum_{i=1}^n \sigma_i W_{ii} \\ \forall i: & -1 \leq W_{ii} \leq 1 \\ & \sum_{i=1}^n W_{ii} \leq n - 2. \end{aligned}$$

The optimal solution to this linear program is obtained at a vertex of the polytope defined by the constraints. We now characterize the vertices of the polytope as follows:

Claim 8 *Any vertex of the polytope defined by the constraints of the above linear program satisfies $W_{ii} \in \{+1, -1\}$ for all i , with at least one W_{ii} set to -1 .*

Proof: Any vertex is obtained by setting n of the inequalities to equalities.

Case 1: n of the $-1 \leq W_{ii} \leq 1$ inequalities are tight. Then all $W_{ii} \in \{-1, +1\}$, and to satisfy $\sum_{i=1}^n W_{ii} \leq n - 2$, we must have at least one -1 .

Case 2: $\sum_{i=1}^n W_{ii}$ equals the integer $n - 2$, exactly $n - 1$ of the inequalities $-1 \leq W_{ii} \leq 1$ are tight for say $1 \leq i \leq n - 1$, and the last one is not tight, i.e. $-1 < W_{nn} < 1$. Then for all $1 \leq i \leq n - 1$, we have $W_{ii} \in \{+1, -1\}$, since $\sum_{i=1}^n W_{ii} = n - 2$, an integer, W_{nn} is also an integer, and hence must be zero. But then with $W_{ii} \in \{+1, -1\}$ for $1 \leq i \leq n - 1$ the sum $\sum_{i=1}^{n-1} W_{ii}$ is either $n - 1$ or at most $n - 3$. Thus $\sum_{i=1}^n W_{ii} = n - 2$ can't be satisfied and case 2 does not give any more vertices. ■

With this characterization of the vertices, since the σ_n is the smallest singular value, the optimal vertex for the linear program is the one where $W_{ii} = 1$ for $1 \leq i \leq n - 1$, and $W_{nn} = -1$. Thus the optimum value of the linear program is $\sum_{i=1}^{n-1} \sigma_i - \sigma_n$. Since this is a relaxation to the original problem, this optimum value is only larger than the optimum of the original problem. However, by setting $\mathbf{W} = \text{diag}(1, 1, \dots, 1, -1)$, which is an orthogonal matrix of determinant -1 , we achieve the same value in the original problem as in the relaxed LP, and hence the optimal solution to the original problem is given by this \mathbf{W} . ■

B Proof of Claim 7

For notational convenience, define a “hallucinated” 0-th trial with loss function $L_0(\mathbf{R}) := -\mathbf{N} \bullet \mathbf{R}$ over rotation matrices. With this notation, $\mathbf{R}_t = \arg \min_{\mathbf{R} \in \mathcal{SO}(n)} \sum_{\tau=0}^{t-1} L_t(\mathbf{R})$, for any $t \geq 1$. We prove by induction that for any $T \geq 0$, we have

$$\sum_{t=0}^T L_t(\mathbf{R}_{t+1}) \leq \sum_{t=0}^T L_t(\mathbf{R}_{T+1}).$$

The statement holds trivially for $T = 0$. So assume that it holds for some $T \geq 0$, and now we prove it for $T + 1$. For this, we have

$$\sum_{t=0}^{T+1} L_t(\mathbf{R}_{t+1}) \leq \sum_{t=0}^T L_t(\mathbf{R}_{T+1}) + L_{T+1}(\mathbf{R}_{T+2}) \leq \sum_{t=0}^T L_t(\mathbf{R}_{T+2}) + L_{T+1}(\mathbf{R}_{T+2}) = \sum_{t=0}^{T+1} L_t(\mathbf{R}_{T+2}).$$

Here, the first inequality follows from the induction hypothesis, and the second from the fact that $\mathbf{R}_{T+1} = \arg \min_{\mathbf{R} \in \mathcal{S}\mathcal{O}(n)} \sum_{t=0}^T L_t(\mathbf{R})$. Thus, the induction is complete.

We now continue by using $\mathbf{R}_{T+1} = \arg \min_{\mathbf{R} \in \mathcal{S}\mathcal{O}(n)} \sum_{t=0}^T L_t(\mathbf{R})$ a second time:

$$\sum_{t=0}^T L_t(\mathbf{R}_{t+1}) \leq \sum_{t=0}^T L_t(\mathbf{R}_{T+1}) \leq \sum_{t=0}^T L_t(\mathbf{R}^*).$$

This implies that

$$\sum_{t=1}^T L_t(\mathbf{R}_t) - \sum_{t=1}^T L_t(\mathbf{R}^*) \leq \sum_{t=1}^T (L_t(\mathbf{R}_t) - L_t(\mathbf{R}_{t+1})) - L_0(\mathbf{R}_1) + L_0(\mathbf{R}^*),$$

as required.

Evolution with Drifting Targets

Varun Kanade*
Harvard University
vkanade@fas.harvard.edu

Leslie G. Valiant*
Harvard University
valiant@seas.harvard.edu

Jennifer Wortman Vaughan†
Harvard University
jenn@seas.harvard.edu

Abstract

We consider the question of the stability of evolutionary algorithms to gradual changes, or *drift*, in the target concept. We define an algorithm to be resistant to drift if, for some inverse polynomial drift rate in the target function, it converges to accuracy $1 - \epsilon$ with polynomial resources, and then stays within that accuracy indefinitely, except with probability ϵ at any one time. We show that every evolution algorithm, in the sense of Valiant [20], can be converted using the Correlational Query technique of Feldman [9], into such a drift resistant algorithm. For certain evolutionary algorithms, such as for Boolean conjunctions, we give bounds on the rates of drift that they can resist. We develop some new evolution algorithms that are resistant to significant drift. In particular, we give an algorithm for evolving linear separators over the spherically symmetric distribution that is resistant to a drift rate of $O(\epsilon/n)$, and another algorithm over the more general product normal distributions that resists a smaller drift rate.

The above translation result can be also interpreted as one on the robustness of the notion of evolvability itself under changes of definition. As a second result in that direction we show that every evolution algorithm can be converted to a quasi-monotonic one that can evolve from any starting point without the performance ever dipping significantly below that of the starting point. This permits the somewhat unnatural feature of arbitrary performance degradations to be removed from several known robustness translations.

1 Overview

The evolvability model introduced by Valiant [20] was designed to provide a quantitative theory for studying mechanisms that can evolve in populations of realistic size, in a reasonable number of generations through the Darwinian process of variation and selection. It models evolving mechanisms as functions of many arguments, where the value of a function represents the outcome of the mechanism, and the arguments the controlling factors. For example, the function might determine the expression level of a particular protein given the expression levels of related proteins. Evolution is then modeled as a restricted form of learning from examples, in which the learner observes only the empirical performance of a set of functions that are feasible variants of the current function. The *performance* of a function is defined as its correlation with the *ideal* function, which specifies for every possible circumstance the behavior that is most beneficial in the current environment for the evolving entity.

The evolution process consists of repeated applications of a random variation step followed by a selection step. In the variation step of round i , a polynomial number of variants of the algorithm's current hypothesis r_i are generated, and their performance empirically tested. In the selection step, one of the variants with high performance is chosen as r_{i+1} . An algorithm therefore consists of both a procedure for describing possible variants and as well as a selection mechanism for choosing among the variants. The algorithm succeeds if it produces a hypothesis with performance close to the ideal function using only a polynomial amount of resources (in terms of number of generations and population size).

*This work was supported in part by NSF-CCF-04-27129.

†Vaughan is supported by NSF under grant CNS-0937060 to the CRA for the CIFellows Project. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors alone.

The basic model as defined in Valiant [20] is concerned with the evolution of Boolean functions using representations that are randomized Boolean functions. This has been shown by Feldman [10] to be a highly robust class under variations in definition, as is necessary for any computational model that aims to capture the capabilities and limitations of a natural phenomenon. This model has also been extended to allow for representations with real number values, in which case a range of models arise that differ according to whether the quadratic loss or some other metric is used in evaluating performance [18, 10]. Our interest here remains with the original Boolean model, which is invariant under changes of this metric.

In this paper we consider the issue of stability of an evolution algorithm to gradual changes, or *drift*, in the target or ideal function. Such stability is a desirable property of evolution algorithms that is not explicitly captured in the original definition. We present two main results in this paper. First, for specific evolution algorithms we quantify how resistant they are to drift. Second, we show that evolutionary algorithms can be transformed to stable ones, showing that the evolutionary model is robust also under modifications that require resistance to drift.

The issue of resistance to drift has been discussed informally before in the context of evolution algorithms that are monotone in the sense that their performance is increasing, or at least non-decreasing, at every stage [18, 10]. We shall therefore start by distinguishing among three notions of monotonicity in terms of properties that need to hold with high probability: (i) quasi-monotonic, where for any ϵ the performance never goes more than ϵ below that of the starting hypothesis r_0 , (ii) monotonic, where the performance never goes below that of r_0 , and (iii) strictly monotonic, where performance increases by at least an inverse polynomial amount at each step. Definition (ii) is essentially Feldman's [10] and definition (iii) is implicit in Michael [18].

We define a notion of an evolution algorithm being stable to drift in the sense that for some inverse polynomial amount of drift, using only polynomial resources, the algorithm will converge to performance $1 - \epsilon$, and will stay with such high performance in perpetuity in the sense that at every subsequent time, except with probability ϵ , its performance will be at least $1 - \epsilon$.

As our main result demonstrating the robustness of the evolutionary model itself, we show, through the simulation of query learning algorithms [9], that for every distribution D , every function class that is evolvable in the original definition, is also evolvable by an algorithm that is both (i) quasi-monotonic, and (ii) stable to some inverse polynomial amount of drift. While the definitions allow any small enough inverse polynomial drift rate, they require good performance in perpetuity, and with the same representation class for all ϵ . Some technical complications arise as a result of the latter two requirements.

As a vehicle for studying the stability of specific algorithms, we show that there are natural evolutionary algorithms for linear separators over symmetric distributions and over the more general product normal distributions. Further we formulate a general result that states that for any strictly monotonic evolution algorithm, where the increase in performance at every step is defined by an inverse polynomial b , one can determine upper bounds on the polynomial parameters of the evolution algorithm, namely those that bound the generation numbers, population sizes, and sample sizes, and also a lower bound on the drift that can be resisted. We illustrate the usefulness of this formulation by applying it to show that our algorithms for linear separators can resist a significant amount of drift. We also apply it to existing algorithms for evolving conjunctions over the uniform distribution, with or without negations. We note that the advantages of evolution algorithms that use natural representations, over those obtained through simulations of query learning algorithms, may be quantified in terms of how moderate the degrees are of the polynomials that bound the generation number, population size, sample size and (inverse) drift rate of these algorithms. These results appear in Sections 6 and 7 and may be read independently of Section 5.

All omitted details and proofs appear in a longer version of this paper, available online [15].

2 The Computational Model of Evolution

In this section, we provide an overview of the original computational model of evolution (Valiant [20], where further details can be found). Many of these notions will be familiar to readers who are acquainted with the PAC model of learning [19].

2.1 Basic Definitions

Let \mathcal{X} be a space of examples. A *concept class* \mathcal{C} over \mathcal{X} is a set of functions mapping elements in \mathcal{X} to $\{-1, 1\}$. A *representation class* \mathcal{R} over \mathcal{X} consists of a set of (possibly randomized) functions from \mathcal{X} to $\{-1, 1\}$ described in a particular language. Throughout this paper, we think of \mathcal{C} as the class of functions from which the ideal target f is selected, and \mathcal{R} as a class of representations from which the evolutionary algorithm chooses an r to approximate f . We consider only classes of

representations that can be evaluated efficiently, that is, classes \mathcal{R} such that for any $r \in \mathcal{R}$ and any $x \in \mathcal{X}$, $r(x)$ can be evaluated in time polynomial in the size of x .

We associate a *complexity parameter* n with \mathcal{X} , \mathcal{C} , and \mathcal{R} . This parameter indicates the number of dimensions of each element in the domain. For example, we might define \mathcal{X}_n to be $\{-1, 1\}^n$, \mathcal{C}_n to be the class of monotone conjunctions over n variables, and \mathcal{R}_n to be the class of monotone conjunctions over n variables with each conjunction represented as a list of variables. Then $\mathcal{C} = \{\mathcal{C}_n\}_{n=1}^{\infty}$ and $\mathcal{R} = \{\mathcal{R}_n\}_{n=1}^{\infty}$ are really ensembles of classes.¹ Many of our results depend on this complexity parameter n . However, we drop the subscripts when the meaning is clear from context.

The *performance* of a representation r with respect to the ideal target f is measured with respect to a distribution \mathcal{D} over examples. This distribution represents the relative frequency with which the organism faces each set of conditions in \mathcal{X} . Formally, for any pair of functions $f : \mathcal{X} \rightarrow \{-1, 1\}$, $r : \mathcal{X} \rightarrow \{-1, 1\}$, and distribution \mathcal{D} over \mathcal{X} , we define the *performance* of r with respect to f as

$$\text{Perf}_f(r, \mathcal{D}) = \mathbb{E}_{x \sim \mathcal{D}}[f(x)r(x)] = 1 - 2\text{err}_{\mathcal{D}}(f, r),$$

where $\text{err}_{\mathcal{D}}(f, r) = \Pr_{x \sim \mathcal{D}}(f(x) \neq r(x))$ is the 0/1 error between f and r . The performance thus measures the correlation between f and r and is always between -1 and 1 .

A new mutation is selected after each round of variation based in part on the observed fitness of the variants, i.e., their empirical correlations with the target on a polynomial number of examples. Formally, the *empirical performance* of r with respect to f on a set of examples x_1, \dots, x_s chosen independently according to \mathcal{D} is a random variable defined as $(1/s) \sum_{i=1}^s f(x_i)r(x_i)$.

We denote by ϵ an *accuracy parameter* specifying how close to the ideal target a representation must be to be considered good. A representation r is a good approximation of f if $\text{Perf}_f(r, \mathcal{D}) \geq 1 - \epsilon$ (or equivalently, if $\text{err}_{\mathcal{D}}(f, r) \leq \epsilon/2$). We allow the evolution algorithm to use resources that are polynomial in both $1/\epsilon$ and the dimension n .

2.2 Model of Variation and Selection

An *evolutionary algorithm* \mathcal{E} determines at each round i which set of mutations of the algorithm's current hypothesis r_{i-1} should be evaluated as candidates for r_i , and how the selection will be made. The algorithm $\mathcal{E} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$ is specified by the following set of components:

- The *representation class* $\mathcal{R} = \{\mathcal{R}_n\}_{n=1}^{\infty}$ specifies the space of representations over \mathcal{X} from which the algorithm may choose functions r to approximate the target f .
- The (possibly randomized) function $\text{Neigh}(r, \epsilon)$ specifies for each $r \in \mathcal{R}_n$ the set of representations $r' \in \mathcal{R}_n$ into which r can randomly mutate. This set of representations is referred to as the *neighborhood* of r . For all r and ϵ , it is required that $r \in \text{Neigh}(r, \epsilon)$ and that the size of the neighborhood is upper bounded by a polynomial.
- The function $\mu(r, r', \epsilon)$ specifies for each $r \in \mathcal{R}_n$ and each $r' \in \text{Neigh}(r, \epsilon)$ the probability that r mutates into r' . It is required that for all r and ϵ , for all $r' \in \text{Neigh}(r, \epsilon)$, $\mu(r, r', \epsilon) \geq 1/p(n, 1/\epsilon)$ for a polynomial p .
- The function $t(r, \epsilon)$, referred to as the *tolerance* of \mathcal{E} , determines the difference in performance that a mutation in the neighborhood of r must exhibit in order to be considered a “beneficial”, “neutral”, or “deleterious” mutation. The tolerance is required to be bounded from above and below, for all representations r , by a pair of inverse polynomials in n and $1/\epsilon$.
- Finally, the function $s(r, \epsilon)$, referred to as the *sample size*, determines the number of examples used to evaluate the empirical performance of each $r' \in \text{Neigh}(r, \epsilon)$. The sample size must also be polynomial in n and $1/\epsilon$.

The functions Neigh , μ , t , and s must all be computable in time polynomial in n and $1/\epsilon$.

We are now ready to describe a single round of the evolution process. For any ideal target $f \in \mathcal{C}$, distribution \mathcal{D} , evolutionary algorithm $\mathcal{E} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$, accuracy parameter ϵ , and representation r_{i-1} , the *mutator* $M(f, \mathcal{D}, \mathcal{E}, \epsilon, r_{i-1})$ returns a random mutation $r_i \in \text{Neigh}(r_{i-1}, \epsilon)$ using the following selection procedure. First, for each $r \in \text{Neigh}(r_{i-1}, \epsilon)$, the mutator computes the empirical performance of r with respect to f on a sample of size s .² Call this $v(r)$. Let

$$\text{Bene} = \{r \mid r \in \text{Neigh}(r_{i-1}, \epsilon), v(r) \geq v(r_{i-1}) + t(r_{i-1}, \epsilon)\}$$

¹As in the PAC model, n should additionally upper bound the size of representation of the function to be learned, but for brevity we shall omit this aspect here.

²We assume a single sample is used to evaluate the performance of all neighbors and r_{i-1} , but one could interpret the model as using independent samples for each representation. This would not change our results.

be the set of “beneficial” mutations and

$$\text{Neut} = \{r \mid r \in \text{Neigh}(r_{i-1}, \epsilon), |v(r) - v(r_{i-1})| < t(r_{i-1}, \epsilon)\}$$

be the set of “neutral” mutations. If at least one beneficial mutation exists, then a mutation r is chosen from **Bene** as the survivor r_i with relative probability $\mu(r_{i-1}, r, \epsilon)$. If no beneficial mutation exists, then a mutation r is chosen from **Neut** as the survivor r_i , again with probability proportional to $\mu(r_{i-1}, r, \epsilon)$. Notice that, by definition, r_{i-1} is always a member of **Neut**, and hence a neutral mutation is guaranteed to exist.

2.3 Putting It All Together

A concept class \mathcal{C} is said to be evolvable by algorithm \mathcal{E} over distribution \mathcal{D} if for every target $f \in \mathcal{C}$, starting at any $r_0 \in \mathcal{R}$, the sequence of mutations defined by \mathcal{E} converges in polynomial time to a representation r whose performance with respect to f is close to 1. This is formalized as follows.

Definition 1 (Evolvability [20]) *For a concept class \mathcal{C} , distribution \mathcal{D} , and evolutionary algorithm $\mathcal{E} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$, we say that \mathcal{C} is evolvable over \mathcal{D} by \mathcal{E} if there exists a polynomial $g(n, 1/\epsilon)$ such that for every $n \in \mathbb{N}$, $f \in \mathcal{C}_n$, $r_0 \in \mathcal{R}_n$, and $\epsilon > 0$, with probability at least $1 - \epsilon$, a sequence r_0, r_1, r_2, \dots generated by setting $r_i = M(f, \mathcal{D}, \mathcal{E}, \epsilon, r_{i-1})$ for all i satisfies $\text{Perf}_f(r_{g(n, 1/\epsilon)}, \mathcal{D}) \geq 1 - \epsilon$.*

We say that the class \mathcal{C} is *evolvable over \mathcal{D}* if there exists a valid evolution algorithm $\mathcal{E} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$ such that \mathcal{C} is evolvable over \mathcal{D} by \mathcal{E} . The polynomial $g(n, 1/\epsilon)$, referred to as the *generation polynomial*, is an upper bound on the number of generations required for the evolution process to converge. If the above definition holds only for a particular value (or set of values) for r_0 , then we say that \mathcal{C} is evolvable *with initialization*.

2.4 Alternative Models

Various alternative formulations of the basic computational model of evolution described here have been studied. Many have been proved equivalent to the basic model in the sense that any concept class \mathcal{C} evolvable in the basic model is evolvable in the alternative model and vice versa. Here we briefly discuss some of the variations that have been considered.

The performance measure $\text{Perf}_f(r, \mathcal{D})$ is defined in terms of the 0/1 loss. Alternative performance measures based on squared loss or other loss functions have been studied in the context of evolution [10, 11, 18]. However, these alternative measures are identical to the original when f and r are (possibly randomized) binary functions, as we have assumed. (When the model is extended to allow real-valued function output, evolvability with a performance measure based on any non-linear loss function is strictly more powerful than evolvability with the standard correlation-based performance measure [10]. We do not consider that extension in this work.)

Alternate rules for determining how a mutation is selected have also been considered. In particular, Feldman [10] showed that evolvability using a selection rule that always chooses among the mutations with the highest or near highest empirical performance in the neighborhood is equivalent to evolvability with the original selection rule based on the classes **Bene** and **Neut**. He also discussed the performance of “smooth” selection rules, in which the probability of a given mutation surviving is a smooth function of its original frequency and the performance of mutations in the neighborhood.

Finally, Feldman [9, 10] showed that *fixed-tolerance* evolvability, in which the tolerance t is a function of only n and $1/\epsilon$ but not the representation r_{i-1} , is equivalent to the basic model.

3 Notions of Monotonicity

Feldman [10, 11] introduced the notion of monotonic evolution in the computational model described above. His notion of monotonicity, restated here in Definition 2, requires that with high probability, the performance of the current representation r_i never drops below the performance of the initial representation r_0 during the evolution process.

Definition 2 (Monotonic Evolution) *An evolution algorithm \mathcal{E} monotonically evolves a class \mathcal{C} over a distribution \mathcal{D} if \mathcal{E} evolves \mathcal{C} over \mathcal{D} and with probability at least $1 - \epsilon$, for all $i \leq g(n, 1/\epsilon)$, $\text{Perf}_f(r_i, \mathcal{D}) \geq \text{Perf}_f(r_0, \mathcal{D})$, where $g(n, 1/\epsilon)$ and r_0, r_1, \dots are defined as in Definition 1.*

When explicit initialization of the starting representation r_0 is prohibited, this is equivalent to requiring that $\text{Perf}_f(r_i, \mathcal{D}) \geq \text{Perf}_f(r_{i-1}, \mathcal{D})$ for all $i \leq g(n, 1/\epsilon)$. In other words, it is equivalent to requiring that with high probability, performance never decreases during the evolution process.

(Feldman showed that if representations may produce real-valued output and an alternate performance measure based on squared loss is considered, then any class \mathcal{C} that is efficiently SQ learnable over a known, efficiently samplable distribution \mathcal{D} is monotonically evolvable over \mathcal{D} .)

A stronger notion of monotonicity was used by Michael [18], who, in the context of real-valued representations and quadratic loss functions, developed an evolution algorithm for learning 1-decision lists in which only beneficial mutations are allowed. In this spirit, we define the notion of *strict* monotonic evolution, which requires a significant (inverse polynomial) performance increase at every round of evolution until a representation with sufficiently high performance is found.

Definition 3 (Strict Monotonic Evolution) *An evolution algorithm \mathcal{E} strictly monotonically evolves a class \mathcal{C} over a distribution \mathcal{D} if \mathcal{E} evolves \mathcal{C} over \mathcal{D} and, for a polynomial m , with probability at least $1 - \epsilon$, for all $i \leq g(n, 1/\epsilon)$, either $\text{Perf}_f(r_{i-1}, \mathcal{D}) \geq 1 - \epsilon$ or $\text{Perf}_f(r_i, \mathcal{D}) \geq \text{Perf}_f(r_{i-1}, \mathcal{D}) + 1/m(n, 1/\epsilon)$, where $g(n, 1/\epsilon)$ and r_0, r_1, \dots are defined as in Definition 1.*

Below we show that a class \mathcal{C} is strictly monotonically evolvable over a distribution \mathcal{D} using representation class \mathcal{R} if and only if it is possible to define a neighborhood function satisfying the property that for any $r \in \mathcal{R}$ and $f \in \mathcal{C}$, if $\text{Perf}_f(r, \mathcal{D})$ is not already near optimal, there exists a neighbor r' of r such that r' has a noticeable (again, inverse polynomial) performance improvement over r . We call such a neighborhood function *strictly beneficial*. The idea of strictly beneficial neighborhood functions plays an important role in developing our results in Sections 6 and 7. Feldman [11] uses a similar notion to show monotonic evolution under square loss.

Definition 4 (Strictly Beneficial Neighborhood Function) *For a concept class \mathcal{C} , distribution \mathcal{D} , and representation class \mathcal{R} , we say that a (possibly randomized) function Neigh is a strictly beneficial neighborhood function if the size of $\text{Neigh}(r, \epsilon)$ is upper bounded by a polynomial $p(n, 1/\epsilon)$, and there exists a polynomial $b(n, 1/\epsilon)$ such that for every $n \in \mathbb{N}$, $f \in \mathcal{C}_n$, $r \in \mathcal{R}_n$, and $\epsilon > 0$, if $\text{Perf}_f(r, \mathcal{D}) < 1 - \epsilon/2$, then there exists a $r' \in \text{Neigh}(r, \epsilon)$ such that $\text{Perf}_f(r', \mathcal{D}) \geq \text{Perf}_f(r, \mathcal{D}) + 1/b(n, 1/\epsilon)$. We refer to $b(n, 1/\epsilon)$ as the benefit polynomial.*

Lemma 5 *For any concept class \mathcal{C} , distribution \mathcal{D} , and representation class \mathcal{R} , if Neigh is a strictly beneficial neighborhood function for \mathcal{C} , \mathcal{D} , and \mathcal{R} , then there exist valid functions μ , t , and s such that \mathcal{C} is strictly monotonically evolvable over \mathcal{D} by $\mathcal{E} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$. If a concept class \mathcal{C} is strictly monotonically evolvable over \mathcal{D} by $\mathcal{E} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$, then Neigh is a strictly beneficial neighborhood function for \mathcal{C} , \mathcal{D} , and \mathcal{R} .*

The proof of the second half of the lemma is immediate; the definition of strictly monotonic evolvability requires that for any initial representation $r_0 \in \mathcal{R}$, with high probability either $\text{Perf}_f(r_0, \mathcal{D}) \geq 1 - \epsilon/2$ or $\text{Perf}_f(r_1, \mathcal{D}) \geq \text{Perf}_f(r_0, \mathcal{D}) + 1/m(n, 2/\epsilon)$ for a polynomial m . Thus if $\text{Perf}_f(r_0, \mathcal{D}) < 1 - \epsilon/2$ there must exist an r_1 in the neighborhood of r_0 such that $\text{Perf}_f(r_1, \mathcal{D}) \geq \text{Perf}_f(r_0, \mathcal{D}) + 1/m(n, 2/\epsilon)$. The key idea behind the proof of the first half is to show that it is possible to set the tolerance $t(r, \epsilon)$ in such a way that with high probability, Bene is never empty and there is never a representation in Bene with performance too much worse than that of the beneficial mutation guaranteed by the definition of the strictly beneficial neighborhood function. This implies that the mutation algorithm is guaranteed to choose a new representation with a significant increase in performance at each round.

Finally, we define quasi-monotonic evolution. This is similar to the monotonic evolution, except that the performance is allowed to go slightly below that of r_0 . In Section 5.7, we show that this notion can be made universal, in the sense that every evolvable class is also evolvable quasi-monotonically.

Definition 6 (Quasi-Monotonic Evolution) *An evolution algorithm quasi-monotonically evolves a class \mathcal{C} over \mathcal{D} if \mathcal{E} evolves \mathcal{C} over \mathcal{D} and with probability at least $1 - \epsilon$, for all $i \leq g(n, 1/\epsilon)$, $\text{Perf}_f(r_i, \mathcal{D}) \geq \text{Perf}_f(r_0, \mathcal{D}) - \epsilon$, where $g(n, 1/\epsilon)$ and r_0, r_1, \dots are defined as in Definition 1.*

4 Resistance to Drift

There are many ways one could choose to formalize the notion of drift resistance. Our formalization is closely related to ideas from the work on tracking drifting concepts in the computational learning literature. The first models of concept drift were proposed around the same time by Helmbold and Long [12] and Kuh et al. [17]. In both of these models, at each time i , an input point x_i is drawn from a fixed but unknown distribution \mathcal{D} and labeled by a target function $f_i \in \mathcal{C}$. It is assumed

that the error of f_i with respect to f_{i-1} on \mathcal{D} is less than a fixed value Δ . Helmbold and Long [12] showed that a simple algorithm that chooses a concept to (approximately) minimize error over recent time steps achieves an average error of $\tilde{O}(\sqrt{\Delta d})$ where d is the VC dimension of \mathcal{C} .³ More general models of drift have also been proposed [2, 3].

Let $f_i \in \mathcal{C}$ denote the ideal function on round i of the evolution process. Following Helmbold and Long [12], we make the assumption that for all i , $\text{err}_{\mathcal{D}}(f_{i-1}, f_i) \leq \Delta$ for some value Δ . This is equivalent to assuming that $\text{Perf}_{f_{i-1}}(f_i, \mathcal{D}) \geq 1 - 2\Delta$. Call a sequence of functions satisfying this condition a Δ -drifting sequence. We make no other assumptions on the sequence of ideal functions.

Definition 7 (Evolvability with Drifting Targets) *For a concept class \mathcal{C} , distribution \mathcal{D} , and evolution algorithm $\mathcal{E} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$, we say that \mathcal{C} is evolvable with drifting targets over \mathcal{D} by \mathcal{E} if there exist polynomials $g(n, 1/\epsilon)$ and $d(n, 1/\epsilon)$ such that for every $n \in \mathbb{N}$, $r_0 \in \mathcal{R}_n$, and $\epsilon > 0$, for any $\Delta \leq 1/d(n, 1/\epsilon)$, and every Δ -drifting sequence f_1, f_2, \dots (with $f_i \in \mathcal{C}_n$ for all i), if r_0, r_1, \dots is generated by \mathcal{E} such that $r_i = M(f_{i-1}, \mathcal{D}, \mathcal{E}, \epsilon, r_{i-1})$, then for all $\ell \geq g(n, 1/\epsilon)$, with probability at least $1 - \epsilon$, $\text{Perf}_{f_\ell}(r_\ell, \mathcal{D}) \geq 1 - \epsilon$. We refer to $d(n, 1/\epsilon)$ as the drift polynomial.*

As in the basic definition, we say that the class \mathcal{C} is *evolvable with drifting targets over \mathcal{D}* if there exists a valid evolution algorithm $\mathcal{E} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$ such that \mathcal{C} is evolvable with drifting targets over \mathcal{D} by \mathcal{E} . The drift polynomial specifies how much drift the algorithm can tolerate.

Our first main technical result, Theorem 8, relates the idea of monotonicity described above to drift resistance by showing that given a strictly beneficial neighborhood function for a class \mathcal{C} , distribution \mathcal{D} , and representation class \mathcal{R} , one can construct a mutation algorithm \mathcal{E} such that \mathcal{C} is evolvable with drifting targets over \mathcal{D} by \mathcal{E} . The tolerance t and sample size s of \mathcal{E} and the resulting generation polynomial g and drift polynomial d directly depend only on the benefit polynomial b as described below. The proof is very similar to the proof of the first half of Lemma 5. Once again the key idea is to show that it is possible to set the tolerance such that with high probability, Bene is never empty and there is never a representation in Bene with performance too much worse than the guaranteed beneficial mutation. This implies that the mutation algorithm is guaranteed to choose a new representation with a significant increase in performance with respect to the previous target f_{i-1} at each round i with high probability. As long as f_{i-1} and f_i are sufficiently close, the chosen representation is also guaranteed to have good performance with respect to f_i .

Theorem 8 *For any concept class \mathcal{C} , distribution \mathcal{D} , and representation class \mathcal{R} , if Neigh is a strictly beneficial neighborhood function for \mathcal{C} , \mathcal{D} , and \mathcal{R} , then there exist valid functions μ , t , and s such that \mathcal{C} is evolvable with drifting targets over \mathcal{D} by $\mathcal{E} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$. In particular, if Neigh is strictly beneficial with benefit polynomial $b(n, 1/\epsilon)$, and $p(n, 1/\epsilon)$ is an arbitrary polynomial upper bound on the size of $\text{Neigh}(r, \epsilon)$, then \mathcal{C} is evolvable with drifting targets over \mathcal{D} with*

- any distributions μ that satisfy $\mu(r, r', \epsilon) \geq 1/p(n, 1/\epsilon)$ for all $r \in \mathcal{R}_n$, ϵ , and $r' \in \text{Neigh}(r, \epsilon)$,
- tolerance function $t(r, \epsilon) = 1/(2b(n, 1/\epsilon))$ for all $r \in \mathcal{R}_n$,
- any generation polynomial $g(n, 1/\epsilon) \geq 16b(n, 1/\epsilon)$,
- any sample size $s(n, 1/\epsilon) \geq 128(b(n, 1/\epsilon))^2 \ln(2p(n, 1/\epsilon)g(n, 1/\epsilon)/\epsilon)$, and
- any drift polynomial $d(n, 1/\epsilon) \geq 16b(n, 1/\epsilon)$, which allows drift $\Delta \leq 1/(16b(n, 1/\epsilon))$.

In Sections 6 and 7, which can be read independent of Section 5, we appeal to this theorem in order to prove that some common concept classes are evolvable with drifting targets with relatively large values of Δ . Using Lemma 5, we also obtain the following corollary.

Corollary 9 *If a concept class \mathcal{C} is strictly monotonically evolvable over \mathcal{D} , then \mathcal{C} is evolvable with drifting targets over \mathcal{D} .*

5 Robustness Results

Feldman [9] proved that the original model of evolvability is equivalent to a restriction of the statistical query model of learning [16] known as learning by correlational statistical queries (CSQ) [5]. We extend Feldman's analysis to show that CSQ learning is also equivalent to both evolvability with drifting targets and quasi-monotonic evolvability, and so the notion of evolvability is robust to these changes in definition. We begin by briefly reviewing the CSQ model.

³Throughout the paper, we use the notation \tilde{O} to suppress logarithmic factors.

5.1 Learning from Correlational Statistical Queries

The *statistical query* (SQ) model was introduced by Kearns [16] and has been widely studied due to its connections to learning with noise [1, 4]. Like the PAC model, the goal of an SQ learner is to produce a hypothesis h that approximates the behavior of a target function f with respect to a fixed but unknown distribution \mathcal{D} . Unlike the PAC model, the learner is not given direct access to labeled examples $\langle x, f(x) \rangle$, but is instead given access to a *statistical query oracle*. The learner submits queries of the form (ψ, τ) to the oracle, where $\psi : \mathcal{X} \times \{-1, 1\} \rightarrow [-1, 1]$ is a query function and $\tau \in [0, 1]$ is a tolerance parameter. The oracle responds to each query with any value v such that $|\mathbb{E}_{x \sim \mathcal{D}}[\psi(x, f(x))] - v| \leq \tau$. An algorithm is said to efficiently learn a class \mathcal{C} in the SQ model if for all $n \in \mathbb{N}$, $\epsilon > 0$, and $f \in \mathcal{C}_n$, and every distribution \mathcal{D}_n over \mathcal{X}_n , the algorithm, given access to ϵ and the SQ oracle for f and \mathcal{D}_n , outputs a polynomially computable hypothesis h in polynomial time such that $\text{err}(f, h) \leq \epsilon$. Furthermore it is required that each query (ψ, τ) made by the algorithm can be evaluated in polynomial time given access to f and \mathcal{D}_n . It is known that any class efficiently learnable in the SQ model is efficiently learnable in the PAC model with label noise [16].

A query (ψ, τ) is called a *correlational statistical query* (CSQ) [5] if $\psi(x, f(x)) = \phi(x)f(x)$ for some function $\phi : \mathcal{X} \rightarrow [-1, 1]$. An algorithm \mathcal{A} is said to efficiently learn a class \mathcal{C} in the CSQ model if \mathcal{A} efficiently learns \mathcal{C} in the SQ model using only correlational statistical queries.

It is useful to consider one additional type of query, the $\text{CSQ}_{>}$ query [9]. A $\text{CSQ}_{>}$ query is specified by a triple (ϕ, θ, τ) , where $\phi : \mathcal{X} \rightarrow [-1, 1]$ is a query function, θ is a threshold, and $\tau \in [0, 1]$ is a tolerance parameter. When presented with such a query, a $\text{CSQ}_{>}$ oracle for target f and distribution \mathcal{D} returns 1 if $\mathbb{E}_{x \sim \mathcal{D}}[\phi(x)f(x)] \geq \theta + \tau$, 0 if $\mathbb{E}_{x \sim \mathcal{D}}[\phi(x)f(x)] \leq \theta - \tau$, and arbitrary value of either 1 or 0 otherwise. Feldman [9] showed that if there exists an algorithm for learning \mathcal{C} over \mathcal{D} that makes CSQs, then there exists an algorithm for learning \mathcal{C} over \mathcal{D} using $\text{CSQ}_{>}$ s of the form (ϕ, θ, τ) where $\theta \geq \tau$ for all queries. Furthermore the number of queries made by this algorithm is at most $O(\log(1/\tau))$ times the number of queries made by the original CSQ algorithm.

5.2 Overview of the Reduction

The construction we present uses Feldman’s simulation [9] repeatedly. Fix a concept class \mathcal{C} and a distribution \mathcal{D} such that \mathcal{C} is learnable over \mathcal{D} in the CSQ model. As mentioned above, this implies that there exists a $\text{CSQ}_{>}$ algorithm \mathcal{A} for learning \mathcal{C} over \mathcal{D} . Let \mathcal{H} be the class of hypotheses from which the output of \mathcal{A} is chosen. In the analysis that follows, we restrict our attention to the case in which \mathcal{A} is deterministic. However, the extension of our analysis to randomized algorithms is straightforward using Feldman’s ideas (see Lemma 4.7 in his paper [9]).

First, we present a high level outline of our reduction. Throughout this section we will use randomized Boolean functions. If $\psi : \mathcal{X} \rightarrow [-1, 1]$ is a real valued function, let Ψ denote the randomized Boolean function such that for every x , $\mathbb{E}[\Psi(x)] = \psi(x)$. It can be easily verified that for any function $\phi(x)$, $\mathbb{E}_{x, \Psi}[\phi(x)\Psi(x)] = \mathbb{E}_x[\phi(x)\psi(x)]$. For the rest of this section, we will abuse notation and simply write real-valued functions in place of the corresponding randomized Boolean functions.

Our representation is of the form $r = (1 - \epsilon/2)h + (\epsilon/2)\Phi$. Here h is a hypothesis from \mathcal{H} and Φ is function that encodes the state of the $\text{CSQ}_{>}$ algorithm that is being simulated. Feldman’s simulation only uses the second part. Our simulation runs in perpetuity, restarting Feldman’s simulation each time it has completed. Since the target functions are drifting over time, if h has high performance with respect to the current target function, it will retain the performance for some time steps in the future, but not forever. During this time, Feldman’s simulation on the Φ part produces a new hypothesis h' which has high performance at the time this simulation is completed. At this time, we will transition to a representation $r' = (1 - \epsilon/2)h' + (\epsilon/2)\Phi$, where Φ is reset to start Feldman’s simulation anew. Thus, although the target drifts, our simulation will continuously run Feldman’s simulation to find a hypothesis that has a high performance with respect to the current target.

The rest of section 5 details the reduction. First, we show how a single run of \mathcal{A} is simulated, which is essentially Feldman’s reduction with minor modifications. Then we discuss how to restart this simulation once it has completed. This requires the addition of certain intermediate states to keep the reduction feasible in the evolution model. We also show that our reduction can be made quasi-monotonic. Finally, we show how all this can be done using a representation class that is independent of ϵ , as is required. This last step is shown in the appendix of the long version [15].

5.3 Construction of the Evolutionary Algorithm

We describe the construction of our evolutionary algorithm \mathcal{E} . Let $\tau = \tau(n, 1/\epsilon)$ be a polynomial lower bound on the tolerance of the queries made by \mathcal{A} when run with accuracy parameter $\epsilon/4$. Without loss of generality, we may assume all queries are made with this tolerance. Let $q = q(n, 1/\epsilon)$

be a polynomial upper bound on the number of queries made by \mathcal{A} , and assume that \mathcal{A} makes exactly q queries (if not, redundant queries can be added). Here, we allow our representation class to be dependent on ϵ . However, this restriction may be removed (cf. Appendix A.7.1 [15]). In the remainder of this section we drop the subscripts n and ϵ , except where there is a possibility of confusion.

Following Feldman's notation, let z denote a bit string of length q which records the oracle responses to the queries made by \mathcal{A} ; that is, the i th bit of z is 1 if and only if the answer to the i th query is 1. Let $|z|$ denote the length of z , z^i the prefix of z of length i , and z_i the i th bit of z . Since \mathcal{A} is deterministic, the i th query made by \mathcal{A} depends only on responses to the previous $i-1$ queries. We denote this query by $(\phi_{z^{i-1}}, \theta_{z^{i-1}}, \tau)$, with $\theta_{z^{i-1}} \geq \tau$, as discussed in Section 5.1. Let h_z denote the final hypothesis output by \mathcal{A} given query responses z . Since we have chosen to simulate \mathcal{A} with accuracy parameter $\epsilon/4$, h_z is guaranteed to satisfy $\text{Perf}_f(h_z, \mathcal{D}) \geq 1 - \epsilon/4$ for any function f for which the query responses in z are valid. Finally, let σ denote the empty string.

For every $i \in \{1, \dots, q\}$ and $z \in \{0, 1\}^i$, we define $\Phi_z = (1/q) \sum_{j=1}^i \mathbb{I}(z_j = 1) \phi_{z^{j-1}}(x)$, where \mathbb{I} is an indicator function that is 1 if its input is true and 0 otherwise. For any $h \in \mathcal{H}$, define $r_\epsilon[h, z] = (1 - \epsilon/2)h(x) + (\epsilon/2)\Phi_z(x)$. Recall that each of these real-valued functions can be treated as a randomized Boolean function as required by the evolution model. The performance of this function, which we use as our basic representation, is mainly determined by the performance of h , but by setting the tolerance parameter low enough, the Φ_z part can learn useful information about the (drifting) targets by simulating \mathcal{A} .

Let $\tilde{R}_\epsilon = \{r_\epsilon[h, z] \mid h \in \mathcal{H}, 0 \leq |z| \leq q-1\}$. The representations in \tilde{R}_ϵ will be used for simulating one round of \mathcal{A} . To reach a state where we can restart the simulation, we will need to add intermediate representations. These are defined below.

Let $tu(n, 1/\epsilon)$ be an upper bound on $\epsilon\theta_{z^i}/(8q)$ for all i and z^i . (This will be a polynomial upper bound on all tolerances t that we define below.) Assume for simplicity that $K = 2/tu(n, 1/\epsilon)$ is an integer. Let $w_0 = r_\epsilon[h, z]$, for some $h \in \mathcal{H}$ and $|z| = q$ (w_0 depends on h and z , but to keep notation simple we will avoid subscripts). For $k = 1, \dots, K$, define $w_k = (1 - k(tu(n, 1/\epsilon)/2))w_0$. Notice that $w_K = \mathbf{0}$, where $\mathbf{0}$ is a function that can be realized by a randomized function that ignores its input and predicts $+1$ or -1 randomly. Let $W_\epsilon = \{w_i \mid w_0 = r_\epsilon[h, z], h \in \mathcal{H}, |z| = q, i \in \{0, \dots, K\}\}$. Finally define $\mathcal{R}_\epsilon = \tilde{R}_\epsilon \cup W_\epsilon$. For every representation $r_\epsilon[h, z] \in \tilde{R}_\epsilon$, we set

- $\text{Neigh}(r_\epsilon[h, z], \epsilon) = \{r_\epsilon[h, z], r_\epsilon[h, z0], r_\epsilon[h, z1]\}$,
- $\mu(r_\epsilon[h, z], r_\epsilon[h, z], \epsilon) = \eta$ and $\mu(r_\epsilon[h, z], r_\epsilon[h, z0], \epsilon) = \mu(r_\epsilon[h, z], r_\epsilon[h, z1], \epsilon) = (1 - \eta)/2$,
- $t(r_\epsilon[h, z], \epsilon) = \epsilon\theta_{z^i}/(8q)$.

For the remaining representations $w_k \in W_\epsilon$, with $w_0 = r_\epsilon[h, z]$, we set

- $\text{Neigh}(w_K, \epsilon) = \{w_K, r_\epsilon[\mathbf{0}, \sigma]\}$ and $\text{Neigh}(w_k, \epsilon) = \{w_k, w_{k+1}, r_\epsilon[h_{z, \epsilon}, \sigma]\}$ for all $k < K$,
- $\mu(w_K, w_K) = \eta$ and $\mu(w_K, r_\epsilon[\mathbf{0}, \sigma]) = 1 - \eta$, and $\mu(w_k, w_k, \epsilon) = \eta^2$, $\mu(w_k, w_{k+1}, \epsilon) = \eta - \eta^2$, and $\mu(w_k, r_\epsilon[h_{z, \epsilon}, \sigma]) = 1 - \eta$ for all $k < K$,
- $t(w_k, \epsilon) = tu(n, 1/\epsilon)$.

Finally, let $\eta = \epsilon/(4q + 2K)$, $\tau' = \min\{(\epsilon\tau)/(2q), tu(n, 1/\epsilon)/8\}$, and $s = 1/(2(\tau')^2) \log((6q + 3K)/\epsilon)$. Let $\mathcal{E} = (\mathcal{R}_\epsilon, \text{Neigh}, \mu, t, s)$ with components defined as above. We show that \mathcal{E} evolves \mathcal{C} over \mathcal{D} tolerating drift of $\Delta = (\epsilon\tau)/(4q + 2K + 2)$. This value of drift, while small, is an inverse polynomial in n and $1/\epsilon$ as required. The point to note is that the evolutionary algorithm runs perpetually, while still maintaining high performance on any given round with high probability.

For any representation r , we denote by LPE the union of the low probability events that some estimates of performance are not within τ' of their true value, or that a mutation with relative probability less than 2η (either in Bene or Neut) is selected over other mutations.

5.4 Simulating the CSQ_> Algorithm for Drifting Targets

We now show that it is possible to simulate a CSQ_> algorithm using an evolution algorithm \mathcal{E} even when the target is drifting. However, if we simulate a query (ϕ, θ, τ) on round i , there is no guarantee that the answer to this query will remain valid in future rounds. The following lemma shows that by lowering the tolerance of the simulated query below the tolerance that is actually required by the CSQ_> algorithm, we are able to generate a sequence of query answers that remain valid over many rounds. Specifically, it shows that if v is a valid response for the query $(\phi, \theta, \tau/2)$ with respect to f_i , then v is also a valid response for the query (ϕ, θ, τ) with respect to f_j for any $j \in [i - \tau/(2\Delta), i + \tau/(2\Delta)]$.

Lemma 10 *Let f_1, f_2, \dots be a Δ -drifting sequence with respect to the distribution \mathcal{D} over \mathcal{X} . For any tolerance τ , any threshold θ , any indices i and j such that $|i - j| \leq \tau/(2\Delta)$, and any function*

$\phi : \mathcal{X} \rightarrow [-1, 1]$, if $\mathbb{E}_{x \sim \mathcal{D}}[\phi(x)f_j(x)] \geq \theta + \tau$, then $\mathbb{E}_{x \sim \mathcal{D}}[\phi(x)f_i(x)] \geq \theta + \tau/2$. Similarly, if $\mathbb{E}_{x \sim \mathcal{D}}[\phi(x)f_j(x)] \leq \theta - \tau$, then $\mathbb{E}_{x \sim \mathcal{D}}[\phi(x)f_i(x)] \leq \theta - \tau/2$.

We say that a string z is *consistent* with a target function f , if for all $1 \leq i \leq |z|$, z_i is a valid response to the query $(\phi_{z^{i-1}}, \theta_{z^{i-1}}, \tau)$, with respect to f . Suppose that the algorithm \mathcal{E} starts with representation $r_0 = r_\epsilon[h, \sigma]$. (Recall that σ denotes the empty string.) The following lemma shows that after q time steps, with high probability it will reach a representation $r_\epsilon[h, z]$ where $|z| = q$ and z is consistent with the target function f_q , implying that z is a proper simulation of \mathcal{A} on f_q .

Lemma 11 *If $\Delta \leq \tau/(2q)$, then for any Δ -drifting sequence f_0, f_1, \dots, f_q , if r_0, r_1, \dots, r_q is the sequence of representations of \mathcal{E} starting at $r_0 = r_\epsilon[h, \sigma]$, and if the LPE does not occur for q rounds, then $r_q = r_\epsilon[h, z]$ where $|z| = q$ and z is consistent with f_q .*

The proof uses the following ideas: If the LPE does not occur, there are no mutations of the form $r \rightarrow r$, so the length of z increases by 1 every round, and also all estimates of performance are within τ' of their true value. When this is the case, and after observing that $r_\epsilon[h, z^i 0]$ is always neutral, it is possible to show that for any round i , (i) if $r_\epsilon[h, z^i 1]$ is beneficial, then 1 is a valid answer to the i th query with respect to f_i , (ii) if $r_\epsilon[h, z^i 1]$ is deleterious then 0 is a valid answer for the i th query with respect to f_i , and (c) if $r_\epsilon[h, z^i 1]$ is neutral, then both 0 and 1 are valid answers to the i th query. This implies that z_{i+1} is always a valid answer to the i th query with respect to f_i , and by Lemma 10, with respect to f_q .

5.5 Restarting the Simulation

We now discuss how to restart Feldman's simulation once it completes. Suppose we are in a representation of the form $r_\epsilon[h, z]$, where $|z| = q$, and z is consistent with the current target function f . Then if h_z is the hypothesis output by \mathcal{A} using query responses in z , we are guaranteed that (with high probability) $\text{Perf}_f(h_z, \mathcal{D}) \geq 1 - \epsilon/4$. At this point, we would like the algorithm to choose a new representation $r_\epsilon[h_z, \sigma]$, where σ is the empty string. The intuition behind this move is as follows. The performance of $r_\epsilon[h_z, \sigma]$ is guaranteed to be high (and to remain high for many generations) because much of the weight is on the h_z term. Thus we can use the second term (Φ_σ) to restart the learning process. After q more time steps have passed, it may be the case that the performance of h_z is no longer as high with respect to the new target, but the simulated algorithm will have already found a different hypothesis that does have high performance with respect to this new target.

There is one tricky aspect of this approach. In some circumstances, we may need to restart the simulation by moving from $r_\epsilon[h, z]$ to $r_\epsilon[h_z, \sigma]$ even though z is *not* consistent with f . This situation can arise for two reasons. First, we might be near the beginning of the evolution process when \mathcal{E} has not had enough generations to correctly determine the query responses (starting state may be $r_\epsilon[h, z_0]$ where z_0 has wrong answers). Second, there is some small probability of failure on any given round and we would like the evolutionary algorithm to recover from such failures smoothly. In either case, to handle the situation in which h_z may have performance below zero (or very close), we will also allow $r_\epsilon[h, z]$ to mutate to $r_\epsilon[\mathbf{0}, \sigma]$.

The required changes from $r_\epsilon[h, z]$ to either $r_\epsilon[h_z, \sigma]$ or $r_\epsilon[\mathbf{0}, \sigma]$ described above may be deleterious. To handle this, we employ a technique of Feldman [9], where we first decrease the performance gradually (through neutral mutations) until these mutations are no longer deleterious. The representations defined in W_ϵ achieve this. The claim is that starting from any representation of the form w_k , we reach either $r_\epsilon[h_z, \sigma]$ or $r_\epsilon[\mathbf{0}, \sigma]$ in at most $K - k + 1$ steps, with high probability. Furthermore, since the probability of moving to $r_\epsilon[h_z, \sigma]$ is very high, this representation will be reached if it is ever a neutral mutation (i.e., the LPE does not happen). Thus, the performance always stays above the performance of $r_\epsilon[h_z, \sigma]$. Lemma 12 formalizes this claim.

Lemma 12 *If $\Delta \leq tu(n, 1/\epsilon)/4$, then for any Δ -drifting sequence f_0, f_1, \dots, f_q , if r_0, r_1, \dots, r_q is the sequence of mutations of \mathcal{E} starting at $r_0 = w_k$, then if the LPE does not happen at any time-step, there exists a $j \leq K - k + 1$ such that $r_j = r_\epsilon[h_{z, \epsilon}, \sigma]$ or $r_j = r_\epsilon[\mathbf{0}, \sigma]$. Furthermore, for all $1 \leq i < j$, $\text{Perf}_{f_i}(r_i, \mathcal{D}) \geq \text{Perf}_{f_i}(r_\epsilon[h_{z, \epsilon}, \sigma], \mathcal{D})$.*

5.6 Equivalence to Evolvability with Drifting Targets

Combining these results, we prove the equivalence between evolvability and evolvability with drifting targets starting from any representation in \mathcal{R}_ϵ . The proof we give here uses the representation class \mathcal{R}_ϵ and therefore assumes that the value of ϵ is known. For the needed generalization to the case where $\mathcal{R} = \cup_\epsilon \mathcal{R}_\epsilon$, Feldman's backsliding trick [9] can be used to first reach a representation with zero performance, and then move to a representation in \mathcal{R}_ϵ . Theorem 13 shows that every concept

class that is learnable using CSQs (and thus every class that is evolvable) is evolvable with drifting targets.

Theorem 13 *If \mathcal{C} is evolvable over distribution \mathcal{D} , then \mathcal{C} is evolvable with drifting targets over \mathcal{D} .*

Proof: Let \mathcal{A} be a CSQ_> algorithm for learning \mathcal{C} over \mathcal{D} with accuracy $\epsilon/4$. \mathcal{A} makes $q = q(n, 1/\epsilon)$ queries of tolerance τ and outputs h satisfying $\text{Perf}_f(h, \mathcal{D}) \geq 1 - \epsilon/4$. Let \mathcal{E} be the evolutionary algorithm derived from \mathcal{A} as described in Section 5.3. Recall that $K = 2/tu(n, 1/\epsilon)$, let $g = 2q + K + 1$. We show that starting from an arbitrary representation $r_0 \in \mathcal{R}_\epsilon$, with probability at least $1 - \epsilon$, $\text{Perf}_{f_g}(r_g, \mathcal{D}) \geq 1 - \epsilon$. This is sufficient to show that for all $\ell \geq g$, with probability at least $1 - \epsilon$, $\text{Perf}_{f_\ell}(r_\ell, \mathcal{D}) \geq 1 - \epsilon$, since we can consider the run of \mathcal{E} starting from $r_{\ell-g}$.

With the setting of parameters as described in Section 5.3, with probability at least $1 - \epsilon$, the LPE does not occur for g time steps, i.e., all estimates are within $\tau' = \min\{(\tau\epsilon)/(2q), tu(n, 1/\epsilon)/8\}$ of their true value and unlikely mutations (those with relative probabilities less than 2η) are not chosen. Thus, we can apply the results of Lemmas 11 and 12. We assume that this is the case for the rest of the proof. When $\Delta = (\epsilon\tau)/(4q + 2K + 2)$, the assumption of Lemmas 11 and 12 hold and we can apply them.

First, we argue that starting from an arbitrary representation, in at most $q + K$ steps, we will have reached a representation of the form $r_\epsilon[h, \sigma]$, for some $h \in \mathcal{H}$. If the start representation is $r_\epsilon[h, z]$ for $|z| \leq q - 1$, then in at most $q - 1$ steps we reach a representation of the form $r_\epsilon[h, z']$ with $|z'| = q$, in which case by Lemma 12, the algorithm will transition to representation $r_\epsilon[h, \sigma]$ in at most $K + 1$ additional steps. Alternately, if the start representation is w_k for $k \in \{0, \dots, K\}$ as defined in Section 5.3, then by Lemma 12, we reach a representation of the form $r_\epsilon[h, \sigma]$ in at most $K + 1$ steps.

Let m be the time step when \mathcal{E} first reaches the representation of the form $r_\epsilon[h, \sigma]$. Then using Lemma 11, $r_{m+q} = r_\epsilon[h, z^*]$, where z^* is consistent with f_{m+q} . Let $h^* = h_{z^*, \epsilon}$ be the hypothesis output by the simulated run of \mathcal{A} . Then $\text{Perf}_{f_{m+q}}(h^*, \mathcal{D}) \geq 1 - \epsilon/4$, and hence $\text{Perf}_{f_{m+q}}(r_\epsilon[h^*, \sigma], \mathcal{D}) \geq 1 - 3\epsilon/4$. For the value of Δ we are using, for all $i \leq g$, $\text{Perf}_{f_i}(r_\epsilon[h^*, \sigma], \mathcal{D}) \geq 1 - \epsilon$.

From such a representation, when all estimates of performance are within τ' of their true value and unlikely mutations (those with relative probability $\leq 2\eta$) do not occur, the performance will remain above $1 - \epsilon$. By Lemma 12, the algorithm will move from $r_{m+q} = r_\epsilon[h, z^*]$ to $r_\epsilon[h^*, \sigma]$ in at most $K + 1$ steps, and during these time steps for any time step i it holds that $\text{Perf}_{f_i}(r_i, \mathcal{D}) \geq \text{Perf}_{f_i}(r_\epsilon[h^*, \sigma], \mathcal{D})$. Once $r_\epsilon[h^*, \sigma]$ is reached, for q steps the representations will be of the form $r_\epsilon[h^*, z]$. For any such time step i , $\text{Perf}_{f_i}(r_i, \mathcal{D}) \geq \text{Perf}_{f_i}(r_\epsilon[h^*, \sigma], \mathcal{D})$. This is because if the answers in z are correct (and they will be since the LPE does not happen at any time step), the term Φ_z is made up of only those functions $\phi_{z^{j-1}}$ for which $z^j = 1$, which are those for which $\phi_{z^{j-1}}$ has a correlation greater than $\theta_{z^{j-1}} - \tau \geq 0$ with the target f_i (using Lemma 10). Since as observed above the performance of $r_\epsilon[h^*, \sigma]$ does not degrade below $1 - \epsilon$ in the time horizon we are interested in $\text{Perf}_{f_i}(r_i, \mathcal{D}) \geq \text{Perf}_{f_i}(r_\epsilon[h^*, \sigma]) \geq 1 - \epsilon$. \blacksquare

5.7 Equivalence to Quasi-Monotonic Evolution

Finally, we show that all evolvable classes are also evolvable quasi-monotonically. In the proof of Theorem 13, we showed that for all $\ell \geq g = 2q + K + 1$, with high probability $\text{Perf}_{f_\ell}(r_\ell, \mathcal{D}) \geq 1 - \epsilon$, so quasi-monotonicity is satisfied trivially. Thus we only need to show quasi-monotonicity for the first g steps. We will use the same construction as defined in Section 5.3, with modifications. However, this assumes that the representation knows ϵ , since now the trick of having the performance slide back to zero would violate quasi-monotonicity. To make the representation class independent of ϵ a more complex construction is needed. Details can be found in the appendix of the long version [15].

Theorem 14 *If \mathcal{C} is evolvable over distribution \mathcal{D} , then \mathcal{C} is quasi-monotonically evolvable over \mathcal{D} with drifting targets.*

6 Evolving Hyperplanes with Drifting Targets

In this section, we present two alternative algorithms for evolving n -dimensional hyperplanes with drifting targets. The first algorithm, which generates the neighbors of a hyperplane by rotating it a small amount in one of $2(n - 1)$ directions, tolerates drift on the order of ϵ/n , but only over spherically symmetric distributions. The second algorithm, which generates the neighbors of a hyperplane by shifting single components of its normal vector, tolerates a smaller drift, but works when the distribution is an unknown product normal distribution. To our knowledge, these are the first positive results on evolving hyperplanes in the computational model of evolution.

Formally, let \mathcal{C}_n be the class of all n -dimensional homogeneous linear separators.⁴ For notational convenience, we reference each linear separator in \mathcal{C}_n by the hyperplane’s n -dimensional unit length normal vector $\mathbf{f} \in \mathbb{R}^n$. For every $\mathbf{f} \in \mathcal{C}_n$ and $\mathbf{x} \in \mathbb{R}^n$, we then have that $\mathbf{f}(\mathbf{x}) = 1$ if $\mathbf{f} \cdot \mathbf{x} \geq 0$, and $\mathbf{f}(\mathbf{x}) = -1$ otherwise. The evolution algorithms we consider in this section use a representation class \mathcal{R}_n also consisting of n -dimensional unit vectors, where $\mathbf{r} \in \mathcal{R}_n$ is the normal vector of the hyperplane it represents.⁵ Then $\mathcal{R} = \{\mathbf{r} \mid \|\mathbf{r}\|_2 = 1\}$. We describe the two algorithms in turn.

6.1 An Evolution Algorithm Based on Rotations

For the rotation-based algorithm, we define the neighborhood function of $\mathbf{r} \in \mathcal{R}_n$ as follows. Let $\{\mathbf{u}^1 = \mathbf{r}, \mathbf{u}^2, \dots, \mathbf{u}^n\}$ be an orthonormal basis for \mathbb{R}^n . This orthonormal basis can be chosen arbitrarily (and potentially randomly) as long as $\mathbf{u}^1 = \mathbf{r}$. Then

$$\text{Neigh}(\mathbf{r}, \epsilon) = \mathbf{r} \cup \{\mathbf{r}' \mid \mathbf{r}' = \cos(\epsilon/(\pi\sqrt{n}))\mathbf{r} \pm \sin(\epsilon/(\pi\sqrt{n}))\mathbf{u}^i, i \in \{2, \dots, n\}\}. \quad (1)$$

In other words, each $\mathbf{r}' \in \text{Neigh}(\mathbf{r}, \epsilon)$ is obtained by rotating \mathbf{r} by an angle of $\epsilon/(\pi\sqrt{n})$ in some direction. The size of this neighbor set is clearly $2n - 1$. We obtain the following theorem.

Theorem 15 *Let \mathcal{C} be the class of homogeneous linear separators, \mathcal{R} be the class of homogeneous linear separators represented by unit length normal vectors, and \mathcal{D} be an arbitrary spherically symmetric distribution. Define Neigh as in Equation 1 and let p be any polynomial satisfying $p(n, 1/\epsilon) \geq 2n - 1$. Then \mathcal{C} is evolvable with drifting targets over \mathcal{D} by algorithm $\mathcal{A} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$ with*

- any distributions μ that satisfy $\mu(r, r', \epsilon) \geq 1/p(n, 1/\epsilon)$ for all $r \in \mathcal{R}_n$, ϵ , and $r' \in \text{Neigh}(r, \epsilon)$,
- tolerance function $t(r, \epsilon) = \epsilon/(\pi^3 n)$ for all $r \in \mathcal{R}_n$,
- any generation polynomial $g(n, 1/\epsilon) \geq 8\pi^3 n/\epsilon$,
- a sample size $s(n, 1/\epsilon) = \tilde{O}(n^2/\epsilon^2)$, and
- any drift polynomial $d(n, 1/\epsilon) \geq 8\pi^3 n/\epsilon$, which allows drift $\Delta \leq \epsilon/(8\pi^3 n)$.

To prove this, we need only to show that Neigh is a strictly beneficial neighborhood function for \mathcal{C} , \mathcal{D} , and \mathcal{R} with $b(n, 1/\epsilon) = \pi^3 n/(2\epsilon)$. The theorem then follows from Theorem 8. The analysis relies on the fact that under any spherically symmetric distribution \mathcal{D} (for example, the uniform distribution over a sphere), $\text{err}_{\mathcal{D}}(\mathbf{u}, \mathbf{v}) = \arccos(\mathbf{u} \cdot \mathbf{v})/\pi$, where $\arccos(\mathbf{u} \cdot \mathbf{v})$ is the angle between \mathbf{u} and \mathbf{v} [6]. This allows us to reason about the performance of one function with respect to another by analyzing the dot product between their normal vectors.

6.2 A Component-Wise Evolution Algorithm

We now describe the alternate algorithm for evolving homogeneous linear separators. The guarantees we achieve are inferior to those described in the previous section. However, this algorithm applies when \mathcal{D} is any unknown product normal distribution (with polynomial variance) over \mathbb{R}^n .

Let r_i and f_i denote the i th components of \mathbf{r} and \mathbf{f} respectively (not the values of the representation and ideal function at round i as in previous sections). The alternate algorithm is based on the following observations. First, whenever there exists some i for which r_i and f_i have different signs and aren’t too close to 0, we can obtain a new representation with a non-trivial increase in performance by flipping the sign of r_i . Second, if there are no beneficial sign flips, if there is some i for which r_i is not too close to f_i , we can obtain a new representation with a significant increase in performance by adjusting r_i a little and renormalizing. The amount we must adjust r_i depends on the standard deviation of \mathcal{D} in the i th dimension, so we must try many values when \mathcal{D} is unknown. Finally, if the above conditions do not hold, then the performance of r is already good enough.

Denote by $\{\mathbf{e}_i\}_{i=1}^n$ the basis of \mathbb{R}^n . Let $\sigma_1, \dots, \sigma_n$ be the standard deviation of the distribution \mathcal{D} in the n dimensions. We assume that $1 \geq \sigma_i \geq (1/n)^k$ for some constant k for all i , and that the algorithm is given access to the value of k , but not the particular values σ_i . We define the neighborhood function as $\text{Neigh}(\mathbf{r}, \epsilon) = N_{\text{fl}} \cup N_{\text{sl}}$, where $N_{\text{fl}} = \{\mathbf{r} - 2r_i \mathbf{e}_i \mid i = 1, \dots, d\}$ is the set of representations obtained by flipping the sign of one component of r , and

$$N_{\text{sl}} = \left\{ \frac{r \pm \frac{j\epsilon^2}{12n^k\sqrt{n}}\mathbf{e}_i}{\left\| r \pm \frac{j\epsilon^2}{12n^k\sqrt{n}}\mathbf{e}_i \right\|_2} \mid i \in \{1, \dots, d\}, j \in \{1, \dots, 4n^k\} \right\}$$

is the set obtained by shifting each component by various amounts. We obtain the following.

⁴A homogeneous linear separator is one that passes through the origin. [6]

⁵Technically we must assume that the representations $\mathbf{r} \in \mathcal{R}_n$ and input points $x \in \mathbb{R}^n$ are expressed to a fixed finite precision so that $\mathbf{r} \cdot \mathbf{x}$ is guaranteed to be computable in polynomial time, but for simplicity, in the analysis that follows, we treat both as simply vectors of real numbers.

Theorem 16 *Let \mathcal{C} be the class of homogeneous linear separators, and \mathcal{R} be the class of homogeneous linear separators represented by unit length normal vectors, and \mathcal{D} be a product normal distribution with (unknown) standard deviations $\sigma_1, \dots, \sigma_n$ such that $1 \geq \sigma_i \geq (1/n)^k$ for all i for a constant k . Define Neigh as above and let p be any polynomial such that $p(n, 1/\epsilon) \geq 8n^{2k+1} + 2n$. Then \mathcal{C} is evolvable with drifting targets over \mathcal{D} by algorithm $\mathcal{A} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$ with*

- any distribution μ satisfying $\mu(r, r', \epsilon) \geq 1/p(n, 1/\epsilon)$ for all $r \in \mathcal{R}_n$ and $r' \in \text{Neigh}(r, \epsilon)$,
- tolerance function $t(r, \epsilon) = \epsilon^6/(288n)$,
- any generation polynomial $g(n, 1/\epsilon) \geq 2304n/\epsilon^6$,
- a sample size $s(n, 1/\epsilon) = \tilde{O}(n^2/\epsilon^{12})$, and
- any drift polynomial $d(n, 1/\epsilon) \geq 2304n/\epsilon^6$, which allows drift $\Delta \leq \epsilon^6/(2304n)$.

The proof formalizes the set of observations described above, using them to show that Neigh is a strictly beneficial neighborhood function for \mathcal{C} , \mathcal{D} , and \mathcal{R} with $b(n, 1/\epsilon) = 144n/\epsilon^6$. The theorem is then an immediate consequence of Theorem 8.

7 Evolving Conjunctions with Drifting Targets

We now show that conjunctions are evolvable with drifting targets over the uniform distribution with a drift of $O(\epsilon^2)$, independent of n . We begin by examining monotone conjunctions and prove that the neighborhood function defined by Valiant [20] is a strictly beneficial neighborhood function with $b(n, 1/\epsilon) = \epsilon^2/9$. Our proof uses techniques similar to those used in the simplified analysis of Valiant's algorithm presented by Diochnos and Turán [8]. By building on ideas from Jacobson [14], we extend this result to show that general conjunctions are evolvable with the same rate of drift.

7.1 Monotone Conjunctions

We represent monotone conjunctions using a representation class \mathcal{R} where each $r \in \mathcal{R}$ is a subset of $\{1, \dots, n\}$ such that $|r| \leq \log_2(3/\epsilon)$, representing the conjunction of the variables x_j for all $j \in r$. We therefore allow the representation class to depend on ϵ in our analysis. This dependence is easy to remove (e.g., using Valiant's technique of allowing an initial phase in which the length of the representation decreases until it is below $\log_2(3/\epsilon)$ [20]), but simplifies presentation.

The neighborhood of a representation r consists of the set of conjunctions that are formed by adding a variable to r , removing a variable from r , and swapping a variable in r with a variable not in r , plus the representation r itself. Formally, define the following three sets of conjunctions: $\mathcal{N}^+(r) = \{r \cup \{j\} | j \notin r\}$, $\mathcal{N}^-(r) = \{r \setminus \{j\} | j \in r\}$, and $\mathcal{N}^\pm(r) = \{r \setminus \{j\} \cup \{k\} | j \in S, k \notin S\}$. The neighborhood $\text{Neigh}(r, \epsilon)$ is then defined as follows. Let $q = \lceil \log_2(3/\epsilon) \rceil$. If r is the empty set, then $\text{Neigh}(r, \epsilon) = \mathcal{N}^+(r) \cup r$. If $0 < |r| < q$, then $\text{Neigh}(r, \epsilon) = \mathcal{N}^+(r) \cup \mathcal{N}^-(r) \cup \mathcal{N}^\pm(r) \cup r$. Finally, if $|r| = q$, then $\text{Neigh}(r, \epsilon) = \mathcal{N}^-(r) \cup \mathcal{N}^\pm(r) \cup r$. Note that the size of the neighborhood is bounded by $1 + n + n^2/4$ in the worst case; the combined size of the sets $\mathcal{N}^+(r)$ and $\mathcal{N}^-(r)$ is at most n , and the size of $\mathcal{N}^\pm(r)$ is at most $n^2/4$. We obtain the following theorem.

Theorem 17 *Let \mathcal{C} be the class of monotone conjunctions, \mathcal{R} be the class of monotone conjunctions of size at most $q = \lceil \log_2(3/\epsilon) \rceil$ represented as subsets of indices, and \mathcal{D} be the uniform distribution. Define Neigh as above and let p be any polynomial satisfying $p(n, 1/\epsilon) \geq 1 + n + n^2/4$. Then \mathcal{C} is evolvable with drifting targets over \mathcal{D} by algorithm $\mathcal{A} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$ with*

- any distributions μ that satisfy $\mu(r, r', \epsilon) \geq 1/p(n, 1/\epsilon)$ for all $r \in \mathcal{R}_n$, ϵ , and $r' \in \text{Neigh}(r, \epsilon)$,
- tolerance function $t(r, \epsilon) = \epsilon^2/18$ for all $r \in \mathcal{R}_n$,
- any generation polynomial $g(n, 1/\epsilon) \geq 144/\epsilon^2$,
- a sample size $s(n, 1/\epsilon) = \tilde{O}(1/\epsilon^2)$, and
- any drift polynomial $d(n, 1/\epsilon) \geq 144/\epsilon^2$, which allows drift $\Delta \leq \epsilon^2/144$.

To prove the theorem, we show that Neigh is a strictly beneficial target function with benefit polynomial $b(n, 1/\epsilon) = 9/\epsilon^2$ and once again appeal to Theorem 8. The proof is then essentially just a case-by-case analysis of the performance of the best $r' \in \text{Neigh}(r, \epsilon)$ for an exhaustive set of conditions on r and f .

7.2 General Conjunctions

Jacobson [14] proposed an extension to the algorithm above that applies to general conjunctions. The key innovation in his algorithm is the addition of a fourth set $\mathcal{N}'(r)$ to the neighborhood or r ,

where each $r' \in \mathcal{N}'(r)$ is obtained by negating a subset of the literals in r . We show here that the drift rate of his construction can be analyzed in a similar way to the monotone case.

We represent general conjunctions using a representation class \mathcal{R} where each $r \in \mathcal{R}$ is a subset of $\{1, \dots, n\} \cup \{-1, \dots, -n\}$ such that $|r| \leq \log_2(3/\epsilon)$. Here each r represents the conjunction of literals x_j for all positive $j \in r$ and negated literals x_{-j} for all negative $j \in r$, and we restrict \mathcal{R} so that it is never the case that both $j \in r$ and $-j \in r$. The dependence of this representation class on ϵ can be removed as before.

As before, the neighborhood of a representation r includes the set of conjunctions that are formed by adding a variable to r , removing a variable from r , and swapping a variable in r with a variable not in r , plus the representation r itself. However, it now also includes a fourth set $\mathcal{N}'(r)$ of all conjunctions that can be obtained by negating a subset of the literals of r . The size of the set $\mathcal{N}'(r)$ is at most $2^q \leq 6/\epsilon$, so by a similar argument to the one above, the size of the neighborhood is bounded by $1 + 2n + n^2 + 6/\epsilon$. We obtain the following theorem.

Theorem 18 *Let \mathcal{C} be the class of conjunctions, \mathcal{R} be the class of conjunctions of at most $q = \lceil \log_2(3/\epsilon) \rceil$ literals represented as above, and \mathcal{D} be the uniform distribution. Define Neigh as above and let p be any polynomial satisfying $p(n, 1/\epsilon) \geq 1 + 2n + n^2 + 6/\epsilon$. Then \mathcal{C} is evolvable with drifting targets over \mathcal{D} by $\mathcal{A} = (\mathcal{R}, \text{Neigh}, \mu, t, s)$ with μ, t, g, s , and d as specified in Theorem 17.*

The proof uses many of the same ideas as the proof of Theorem 17. However, there are a few extra cases that need to be considered. First, if f is a “long” conjunction, and r contains at least one literal that is the negation of a literal in f , then we show that adding another literal to r leads to a significant increase in performance. (If r is already of maximum size, then the performance is already good enough.) Second, we show that if f is “short” and r contains at least one literal that is the negation of a literal in f , then there exists an $r' \in \mathcal{N}'(r)$ with significantly better performance. All other cases are identical to the monotone case.

References

- [1] J. Aslam and S. Decatur. Specification and simulation of statistical query algorithms for efficiency and noise tolerance. *Journal of Computer and System Sciences*, 56(2):191–208, 1998.
- [2] P. L. Bartlett. Learning with a slowly changing distribution. In *COLT 5*, 1992.
- [3] R. D. Barve and P. M. Long. On the complexity of learning from drifting distributions. *Information and Computation*, 138(2):101–123, 1997.
- [4] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003.
- [5] N. H. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2:359–395, 2002.
- [6] S. Dasgupta. Coarse sample complexity bounds for active learning. In *NIPS 18*, 2005.
- [7] S. Dasgupta, A. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. *Journal of Machine Learning Research*, pages 281–299, 2009.
- [8] D. I. Diochnos and G. Turán. On evolvability: The swapping algorithm, product distributions, and covariance. In *Fifth Symposium on Stochastic Algorithms, Foundations and Applications*, 2009.
- [9] V. Feldman. Evolvability from learning algorithms. In *Proceedings of ACM STOC 40*, 2008.
- [10] V. Feldman. Robustness of evolvability. In *COLT 22*, 2009.
- [11] V. Feldman. A complete characterization of statistical query learning with applications to evolvability. In *Proceedings of the IEEE Symposium on Foundation of Computer Science*, 2009.
- [12] D. P. Helmbold and P. M. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14(1):27–46, 1994.
- [13] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.
- [14] B. Jacobson. Personal communication, 2007.
- [15] V. Kanade, L. G. Valiant, and J. W. Vaughan. Evolution with drifting targets. <http://arxiv.org/abs/1005.3566>, 2010.
- [16] M. Kearns. Efficient noise-tolerant learning from statistical queries. *JACM*, 45(6):983–1006, 1998.
- [17] A. Kuh, T. Petsche, and R. Rivest. Incrementally learning time-varying half-planes. In *NIPS 4*, 1991.
- [18] L. Michael. Evolvability via the Fourier transform. *Theoretical Computer Science*, 2010. To appear.
- [19] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
- [20] L. G. Valiant. Evolvability. *Journal of the ACM*, 56(1):1–21, 2009.

Regret Minimization With Concept Drift

Koby Crammer*
The Technion
koby@ee.technion.ac.il

Eyal Even-Dar
Google Research
evendar@google.com

Yishay Mansour†
Tel Aviv University
mansour@cs.tau.ac.il

Jennifer Wortman Vaughan‡
Harvard University
jenn@seas.harvard.edu

Abstract

In standard online learning, the goal of the learner is to maintain an average loss close to the loss of the best-performing function in a fixed class. Classic results show that simple algorithms can achieve an average loss arbitrarily close to that of the best function in retrospect, even when input and output pairs are chosen by an adversary. However, in many real-world applications such as spam prediction and classification of news articles, the best target function may be drifting over time. We introduce a novel model of concept drift in which an adversary is given control of both the distribution over input at each time step and the corresponding labels. The goal of the learner is to maintain an average loss close to the 0/1 loss of the best slowly changing sequence of functions with no more than K large shifts. We provide regret bounds for learning in this model using an (inefficient) reduction to the standard no-regret setting. We then go on to provide and analyze an efficient algorithm for learning d -dimensional hyperplanes with drift. We conclude with some simulations illustrating the circumstances under which this algorithm outperforms other commonly studied algorithms when the target hyperplane is drifting.

1 Introduction

Consider the classical problem of online learning. At each time step, the learner is given a new data instance (for example, an email) and must output a prediction of its label (for example, “spam” or “not spam”). The true label is then revealed, and the learner suffers a loss based on both the label and its prediction. Generally in this setting, the goal of the learner is to achieve an average loss that is “not too big” compared to the loss it would have received if it had always chosen to predict according to the best-performing function from a fixed class \mathcal{F} . It is well-known that as the number of time steps grows, very simple aggregation algorithms are able to achieve an average loss arbitrarily close to that of the best function in retrospect. Furthermore, such guarantees hold even if the input and output pairs are chosen in a fully adversarial manner with no distributional assumptions [6].

Despite the extensive literature on no-regret learning and the impressive guarantees that can be made, competing with the best fixed function is not always good enough. In many real-world applications, the true target function is not fixed, but is slowly changing over time. Consider a classifier designed to identify news articles related to China. Over time, the most relevant topics might drift from the Olympics to exports to finance to human rights. When this drift occurs, the classifier itself must also change in order to remain relevant. Similarly, the very definition of spam is changing over time as spammers become more creative and deviant. Any useful spam filter must evolve to keep up with this drift.

*Some of this research was completed while KC was at University of Pennsylvania. KC is a Horev Fellow, supported by the Taub Foundations.

†This work was supported in part by a grant from the Ministry of Science (grant No. 3-6797), by a grant from the Israel Science Foundation (grant No. 709/09), by grant No. 2008-321 from the United States-Israel Binational Science Foundation (BSF), and by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication reflects the authors’ views only.

‡Some of this research was completed while Vaughan was at Google. Vaughan is supported by NSF under grant CNS-0937060 to the CRA for the CIFellows Project. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors alone.

With such applications in mind, we develop a new theoretical model for regret minimization with concept drift. Here the goal of the algorithm is no longer to compete well with a single function, but to maintain an average loss close to that of the best slowly changing sequence of functions with no more than K large shifts. In order to achieve this goal, it is necessary to restrict the adversary in some way — in classification, if the adversary is given full power over the choice of input and output, it can force any algorithm to suffer a constant regret simply by choosing the direction of drift at random and selecting input points near the decision boundary, even when $K = 0$. To address this problem, early models of drift assumed a fixed input distribution [10] or a joint distribution over input and output that changes slowly over time [1, 15]. More recently, Cavallanti et al. [5] addressed this problem by bounding the number of mistakes made by the algorithm not in terms of the number of mistakes made by the adversary, but in terms of the adversary’s hinge loss. (Recall that the hinge loss would assign a positive loss to observations near the decision boundary even if no error is made.) We take a different approach, requiring more traditional regret bounds in terms of the adversary’s 0/1 loss while still endowing the adversary with a significant amount of power. We allow the adversary to specify not a single point but a distribution over points at each time. The distributions D_t and D_{t+1} at consecutive times need not be close in any usual statistical sense, and can even have disjoint supports. However, the adversary is prevented from choosing distributions that put too much weight on small regions of input space where pairs of “similar” functions disagree.

Our first algorithmic result shows that learning in this model can be reduced to learning in the standard adversarial online setting. Unfortunately, the resulting algorithms are generally not efficient, in some cases requiring updating an exponential number of weights. Luckily, specialized algorithms can be designed for efficiently learning particular function classes. To gain intuition, we start by providing a simple algorithm for learning one-dimensional threshold functions with drift. We then analyze the performance of the Modified Perceptron algorithm of Blum et al. [4], showing that it can be used to efficiently learn d -dimensional hyperplanes with concept drift.

We conclude with some simulations illustrating the circumstances under which the Modified Perceptron outperforms other algorithms when the target hyperplane is drifting. We find that the Modified Perceptron performs best relative to other algorithms when the underlying dimension of the data is small, even if the data is projected into a high dimensional space. When the underlying dimension of the data is large, the standard Perceptron is equally capable of handling drifting targets. This phenomenon is not explained by our theoretical results, and would be an interesting direction for future research.

2 Related Work

The first model of concept drift was proposed by Helmbold and Long [10]. In their model, at each time t , an input point x_t is drawn from a fixed, unknown distribution D and labeled by a target function f_t , where for each t , the probability that f_t and f_{t+1} disagree on the label of a point drawn from D is less than a fixed value Δ . They showed that a simple algorithm achieves an average error of $\tilde{O}((\Delta d)^{1/2})$ where d is the VC dimension of the function class, or $\tilde{O}((\Delta d)^{1/3})$ in the unrealizable setting. Kuh et al. [13, 14] examined a similar model and provided an efficient algorithm for learning two-dimensional half-planes through the origin and intersections of half-planes through the origin.

Bartlett [1] introduced a more general agnostic model of drift. In this model, the sequence of input and output pairs is generated according to a sequence of joint distributions P_1, \dots, P_T , such that for each t , P_t and P_{t+1} have total variation distance less than Δ . It is easy to verify that the earlier drifting model described above is a special case of this model. Long [15] showed that one can achieve a similar error rates of $O((\Delta d)^{1/2})$ (or $O((\Delta d)^{1/3})$ in the unrealizable setting) in this model, and Barve and Long [3] provided additional upper and lower bounds. Freund and Mansour [8] showed that improvements are possible if the joint distribution is changing at a constant rate. Bartlett et al. [2] also studied a variety of drifting settings, including one in which the target may change arbitrarily but only infrequently.

Most of these models assume a fixed or slowly changing input distribution. At the other extreme lie models in which the input points may be chosen in an arbitrary, adversarial manner. Herbster and Warmuth [11] studied a setting in which the time sequence is partitioned into k segments. The goal of the algorithm is to compete with the best expert in each segment for the best segmentation in retrospect. They later studied algorithms for tracking the best linear predictor with drift [12].

Cavallanti et al. [5] analyzed a variant of the Perceptron algorithm for learning d -dimensional hyperplanes with drift. They bounded the number of mistakes made by the algorithm in terms of the hinge loss of the best sequence of hyperplanes and the amount of drift in a fully adversarial setting. As we briefly discuss in Section 3, there is no way to obtain a result such as theirs in a fully adversarial setting if we wish to measure the regret with respect to the 0/1 loss of the drifting sequence rather

than the hinge loss. We have no choice but to limit the power of the adversary in some way. Finally, Hazan and Seshadhri [9] study drift in the more general online convex optimization setting, providing bounds in terms of the maximum regret (to a single optimal point) achieved over any contiguous time interval. This captures the notion of drift because the optimal point can vary across different time intervals.

Our model falls between these two extremes. On the one hand, we make no requirements on how quickly the distribution over input points can change from one time step to the next, and in fact allow the scenario in which the support of D_t and the support of D_{t+1} do not overlap on any points. However, unlike the purely adversarial models, we require that the distribution chosen at each time step not place “too much” weight on points where pairs of nearby functions disagree. This added requirement gives us the ability to produce bounds in terms of 0/1 loss in situations in which it is provably impossible to learn in a fully adversarial setting.

3 A New Model of Drift

Let \mathcal{F} be a hypothesis class mapping elements in a set \mathcal{X} to elements in a set \mathcal{Y} , and let \mathbf{near} be an arbitrary binary relation over elements in \mathcal{F} . For example, if \mathcal{F} is the class of linear separators, we might define $\mathbf{near}(f, f')$ to hold if and only if the weight vectors corresponding to f and f' are sufficiently close to each other. At a high level, the \mathbf{near} relation is meant to encapsulate some notion of similarity between functions. Our model implicitly assumes that it is common for the target to drift from one function to another function \mathbf{near} it from one time step to the next.

We say that a sequence of functions f_1, \dots, f_T is a K -shift legal sequence if $\mathbf{near}(f_t, f_{t+1})$ holds for at least $T - K$ time steps $t < T$. Unlike standard online learning where the goal is to have low regret with respect to the best single function, the goal in our model is to have low regret with respect to the best K -shift legal sequence. Regret is defined in terms of a loss function \mathcal{L} , which is assumed to satisfy the triangle inequality, be bounded in $[0, 1]$, and satisfy $\mathcal{L}(x, x) = 0$ for all x .

In order to achieve this goal, some restrictions must be made. We cannot expect an algorithm to be able to compete in a fully adversarial setting with drift. To understand why, consider for example the problem of online classification with drifting hyperplanes. Here the adversary can force any algorithm to have an average loss of 1/2 by simply randomizing the direction of drift at each time step and choosing input points near the decision boundary, even when $K = 0$. As such, we work in a setting in which the adversary may specify not a single point but a distribution over points. In particular, the adversary may specify any distribution that is “good” in the following precise sense.¹

Definition 1 *A distribution D is λ -good for loss function \mathcal{L} and binary relation \mathbf{near} if for all pairs $f, f' \in \mathcal{F}$ such that $\mathbf{near}(f, f')$, $E_{x \sim D} [\mathcal{L}(f(x), f'(x))] \leq \lambda$.*

For most of this paper, we restrict our attention to the problem of online classification with $\mathcal{Y} = \{+1, -1\}$ and define \mathcal{L} to be 0/1 loss. In this case, D is λ -good if for every $f, f' \in \mathcal{F}$ such that $\mathbf{near}(f, f')$, $\Pr_{x \sim D} (f(x) \neq f'(x)) \leq \lambda$. Restricting the input distribution in this way ensures that the adversary cannot place too much weight on areas of the input space where pairs of \mathbf{near} functions disagree, while still providing the adversary with the power to select arbitrarily different distributions from one time step to the next. Note that the larger the space of \mathbf{near} pairs is, the smaller the set of λ -good distributions, and vice versa. When the \mathbf{near} space is empty, every distribution is λ -good. At the other one extreme, the set of λ -good distributions might be empty (for example, if the function that classifies all points as positive and the function that classifies all points as negative are defined to be \mathbf{near}). We restrict our attention only to triples $(\mathcal{F}, \mathbf{near}, \lambda)$ such that at least one λ -good distribution exists.

The majority of our results hold in the following adversarial setting. Fix a value of λ and definition of \mathbf{near} . At each time $t \in \{1, \dots, T\}$, the learning algorithm chooses a (possibly randomized) hypothesis h_t . The adversary then chooses an arbitrary λ -good distribution D_t and an arbitrary function $f_t \in \mathcal{F}$. The algorithm is presented with a point x_t distributed according to D_t , learns the label $f_t(x_t)$, and receives a loss $\mathcal{L}(h_t(x_t), f_t(x_t))$. Let K be the number of time steps t for which $\mathbf{near}(f_t, f_{t+1})$ does not hold. (We are usually interested in the case in which K is a small constant.) Then by definition, f_1, \dots, f_T is a K -shift legal sequence. The goal of the learning algorithm is to maintain low expected regret with respect to the best K -shift legal sequence (where the expectation is taken with respect to the random sequence of input points and any internal randomization of the algorithm), which is equivalent to maintaining a small expected average loss since a perfect

¹Of course it could be possible to obtain results by restricting the adversary in other ways, such as requiring that points be chosen to respect a minimum margin assumption. However, some restriction is needed.

K -shift legal sequence is guaranteed to exist. The adversary's choice of D_t and f_t may depend on any available historical information, including the sequence of input points x_1, \dots, x_{t-1} , and on the algorithm itself. The algorithm has no knowledge of the number of shifts K or the times at which these shifts occur. We refer to this scenario as the *realizable* setting.

We also briefly consider an *unrealizable* setting in which the adversary is not required to choose f_t on the fly each time step. Instead, the adversary selects an arbitrary λ -good distribution D_t (again for a fixed value of λ) and a distribution over labels y_t conditioned on x_t . The algorithm is presented with a point x_t distributed according to D_t and a label y_t distributed according to the distribution chosen by the adversary and receives a loss of $\mathcal{L}(h_t(x_t), y_t)$. In this setting, the goal is to maintain low expected regret with respect to the best K -shift legal sequence in retrospect for a fixed value of K , where the expectation is taken with respect to the random input sequence, random labels, and any randomization of the algorithm, and the regret is defined as

$$\sum_{t=1}^T \mathcal{L}(h_t(x_t), y_t) - \min_{f_1, \dots, f_T \in \Phi_K} \sum_{t=1}^T \mathcal{L}(f_t(x_t), y_t),$$

where Φ_K is the set of all K -shift legal sequences f_1, \dots, f_T .

Note that unlike the standard online learning setting, we should not expect the regret per round to go to zero, even in the realizable setting. Suppose that the target is known perfectly at some time t . It is still possible for the algorithm to make an error at time $t + 1$ because the target can move. This uncertainty never goes away, even as the number of time steps grows very large, so we should expect to see a dependence on λ in the average regret that does not diminish over time. This inherent uncertainty is the very heart of the drifting problem.

4 A General Reduction

We now provide general upper bounds for learning finite function classes in the model. We show via a simple reduction that it is possible to achieve an expected average per time step regret of $O((\lambda \log N)^{1/3})$, and that this regret can be reduced to $O(\sqrt{\lambda \log N})$ in the realizable setting.² However, the algorithm used is not always efficient when the function class is very large or infinite. The subsequent sections are devoted to efficient algorithms for particular function classes.

The results rely on the following lemma.

Lemma 2 *Let \mathcal{L} be any loss function with $\mathcal{L}(x, x) = 0$ for all x that satisfies the triangle inequality. For any 0-shift legal sequence f_1, \dots, f_ℓ , and any sequence of joint distributions P_1, \dots, P_ℓ over pairs $\{x_1, y_1\}, \dots, \{x_\ell, y_\ell\}$ such that the marginal distributions D_1, \dots, D_ℓ , over x_1, \dots, x_ℓ are λ -good, there exists a function $f \in \mathcal{F}$ such that $\sum_{t=1}^\ell E_{\{x_t, y_t\} \sim P_t} [\mathcal{L}(f(x_t), y_t)] \leq \sum_{t=1}^\ell E_{\{x_t, y_t\} \sim P_t} [\mathcal{L}(f_t(x_t), y_t)] + (\ell - 1)^2 \lambda / 2$.*

Proof: We first show by induction that for any λ -good distribution D and any 0-shift legal sequence f_1, \dots, f_ℓ , $E_{x \sim D} [\mathcal{L}(f_1(x), f_\ell(x))] \leq (\ell - 1)\lambda$. This clearly holds for $\ell = 1$. Suppose that $E_{x \sim D} [\mathcal{L}(f_1(x), f_{\ell-1}(x))] \leq (\ell - 2)\lambda$. Since the functions form a 0-shift legal sequence and D is λ -good, we must have $E_{x \sim D} [\mathcal{L}(f_{\ell-1}(x), f_\ell(x))] \leq \lambda$. By the triangle inequality and linearity of expectation, $E_{x \sim D} [\mathcal{L}(f_1(x), f_\ell(x))] \leq E_{x \sim D} [\mathcal{L}(f_1(x), f_{\ell-1}(x)) + \mathcal{L}(f_{\ell-1}(x), f_\ell(x))] \leq (\ell - 1)\lambda$.

This implies that for any t , $E_{x_t \sim D_t} [\mathcal{L}(f_1(x_t), y_t)] \leq E_{x_t \sim D_t} [\mathcal{L}(f_t(x_t), y_t) + \mathcal{L}(f_1(x_t), f_t(x_t))] \leq E_{x_t \sim D_t} [\mathcal{L}(f_t(x_t), y_t)] + (t - 1)\lambda$. Summing over all t yields the lemma. \blacksquare

The following theorem provides a general upper bound for the unrealizable setting.

Theorem 3 *Let \mathcal{F} be a finite function class of size N and near be any binary relation on \mathcal{F} that yields a non-empty set of λ -good distributions. There exists an algorithm for learning \mathcal{F} that achieves an average expected regret of $O((\lambda \ln N)^{1/3})$ when $T \geq (\ln N)^{1/3} \lambda^{-2/3}$ for any $K \leq \lambda T$, even in the unrealizable setting.*

Proof: Let \mathcal{A} be any regret minimization algorithm for \mathcal{F} that is guaranteed to have regret at most $r(m)$ over m time steps. We can construct an algorithm for learning \mathcal{F} using \mathcal{A} as a black box. The algorithm simply divides the sequence of T time steps into $\lceil T/m \rceil$ consecutive subsequences of length at most m and runs \mathcal{A} on each of these subsequences.

²Here and throughout this paper, we consider the asymptotic behavior of functions as $\lambda \rightarrow 0$ (or equivalently, as $1/\lambda \rightarrow \infty$). This implies, for example, that an error of $O(\sqrt{\lambda})$ is preferred to an error of $O(\lambda^{1/3})$.

The regret of the algorithm with respect to the best function in \mathcal{F} on each subsequence is at most $r(m)$. Furthermore, by Lemma 2, the best function in \mathcal{F} on this subsequence has a regret of no more than $m^2\lambda$ with respect to any legal sequence and thus also the best legal sequence. Combining these facts with the fact that the error can be no more than m on any subsequence yields a bound of

$$\left\lceil \frac{T}{m} \right\rceil \left(r(m) + \frac{m^2\lambda}{2} \right) + Km \leq \left(\frac{T}{m} + 1 \right) \left(r(m) + \frac{m^2\lambda}{2} \right) + Km$$

on the total expected regret of the algorithm with respect to the best K -shift legal sequence.

There exist well-known algorithms for finite classes with regret $r(m) = O(\sqrt{m \ln N})$ [6]. Letting \mathcal{A} be one of these algorithms and setting $m = (\ln N)^{1/3} \lambda^{-2/3}$ yields the bound. ■

The following theorem shows that in the realizable setting, it is possible to obtain an average loss of $O(\sqrt{\lambda \ln N})$ as long as T is sufficiently large and K sufficiently small compared with T . This is an improvement on the previous bound whenever the bound is not trivial, i.e., when $\lambda \ln N < 1$.

Theorem 4 *Let \mathcal{F} be a finite function class of size N and \mathbf{near} be any binary relation on \mathcal{F} such that the set of λ -good distributions is not empty. There exists an algorithm for learning \mathcal{F} in the realizable setting that achieves an expected average per time step loss of $O(\sqrt{\lambda \ln N})$ when $T \geq \sqrt{\ln N / \lambda}$ for any $K \leq T\lambda$.*

Proof Sketch: The proof is nearly identical to the proof of Theorem 3. The only differences are that the regret minimization algorithm employed must guarantee a regret of $O(\sqrt{L^* \ln N} + \ln N)$ where L^* is the loss of the best expert (see Cesa-Bianchi and Lugosi [6] for examples), and m must be set to $\sqrt{\ln N / \lambda}$. The proof then relies on the fact that in the realizable setting, on any period of length m during which no shift occurs, $L^* \leq m^2\lambda$ (from Lemma 2). ■

The results are easily extended to the case in which \mathcal{F} is not finite but has finite VC dimension d , but hold only under certain restricted definitions of \mathbf{near} .

5 Efficient Algorithms for Drifting Thresholds and Hyperplanes

The reductions described in the previous section are not efficient in general and may require maintaining an exponential number of weight for infinite function classes. In this section, we analyze an efficient Perceptron-style algorithm for learning drifting d -dimensional hyperplanes. To promote intuition, we begin by describing and analyzing a simple specialized algorithm for learning one-dimensional thresholds. The analysis of the Perceptron-style algorithm uses similar ideas.

5.1 One-Dimensional Thresholds

We denote by $\tau_t \in [0, 1]$ the threshold corresponding to the target function f_t ; thus $f_t(x) = 1$ if and only if $x \geq \tau_t$. For any two functions f and f' with corresponding thresholds τ and τ' , we say that the relation $\mathbf{near}(f, f')$ holds if and only if $|\tau - \tau'| \leq \gamma$ for some fixed $\gamma \leq \lambda$. By definition, this implies that any λ -good input distribution D_t can place at most weight λ on any interval of width γ .

A simple algorithm can be used to achieve optimal error bounds in the realizable setting. At each time t , the algorithm keeps track of an interval I_t of threshold values corresponding to all functions that could feasibly be the current target if no major shift has recently occurred. When the input point x_t is observed, the algorithm predicts the label selected by the majority of the threshold values in I_t (that is, 0 if x_t is closer to the lower border of I_t and 1 if it is closer to the upper border).

To start, I_1 is initialized to the full interval $[0, 1]$ since any threshold is possible. At each subsequent time t , one of three things happens. If $x_t \notin I_t$, then the entire feasible set agrees on the label of x_t . If the predicted label is correct, then to allow for the possibility of concept drift, the algorithm sets I_{t+1} to be I_t increased by γ on each side. If the predicted label is incorrect, then it must be the case that a shift has recently occurred, and I_{t+1} is reset to the full interval $[0, 1]$. (Note that this can happen at most K times.) On the other hand, if $x_t \in I_t$, then there is disagreement in the shift-free interval about the label of the point and the algorithm learns new information about the current threshold by learning the label. In this case, all infeasible thresholds are removed from the shift-free feasible set and then again γ is added on each side to account for possible concept drift. Namely, if $x_t \in I_t = (a, b)$ then I_{t+1} is either $(a - \gamma, x_t + \gamma)$ or $(x_t - \gamma, b + \gamma)$. The next theorem shows that this algorithm results in error $O(\sqrt{\lambda})$ as long as T is sufficiently large.

Theorem 5 *Let \mathcal{F} be the class of one-dimensional thresholds and let \mathbf{near} be defined as above. The expected average error of the algorithm described above in the realizable setting is no more than $K/T + \sqrt{2(K+1)\lambda/(T\gamma)} + \sqrt{5\lambda}$.*

Proof: Let inc_t be a random variable that is 1 if the label of the input point at time t is inconsistent with all hypotheses in the version space and 0 otherwise; if $\text{inc}_t = 1$, then the algorithm described above makes a mistake at time t and sets I_{t+1} to the full interval $[0, 1]$. Note that $\sum_{t=1}^T \text{inc}_t \leq K$.

Let err_t be a random variable that is 1 if the label of the input at time t is consistent with some hypothesis in the version space but the algorithm makes an error anyway and 0 if this is not the case. For any positive-width interval I and λ -good distribution D , $D(I) \leq \lceil |I|/\gamma \rceil \lambda \leq (|I|/\gamma + 1)\lambda$, where $|I|$ is the length of interval I . Hence, at every time step t , $\Pr(\text{err}_t = 1 | I_t, f_t, D_t) \leq D_t(I_t) \leq |I_t|/\gamma + \lambda$, and so $|I_t| \geq (\gamma/\lambda)\Pr(\text{err}_t = 1 | I_t, f_t, D_t) - \gamma$.

Since the algorithm predicts according to the majority of the (shift-free) feasible set, it eliminates at least half of the hypotheses in this set on each consistent mistake. However, at every time step, the feasible set can grow by γ on each side. Thus,

$$\begin{aligned} \mathbb{E}[|I_{t+1}| | I_t, f_t, D_t] &\leq \Pr(\text{err}_t = 1 | I_t, f_t, D_t) (|I_t|/2 + 2\gamma) \\ &\quad + (1 - \Pr(\text{err}_t = 1 | I_t, f_t, D_t)) (|I_t| + 2\gamma) + \Pr(\text{inc}_t = 1 | I_t, f_t, D_t) \cdot 1 \\ &= |I_t| + 2\gamma - (|I_t|/2)\Pr(\text{err}_t = 1 | I_t, f_t, D_t) + \Pr(\text{inc}_t = 1 | I_t, f_t, D_t) \\ &\leq |I_t| + 5\gamma/2 - (\gamma/(2\lambda))\Pr(\text{err}_t = 1 | I_t, f_t, D_t)^2 + \Pr(\text{inc}_t = 1 | I_t, f_t, D_t), \end{aligned}$$

where the final step follows from the lower bound on $|I_t|$ given above. Taking the expectation of both sides with respect to $\{I_t, f_t, D_t\}$ gives us that for any t ,

$$\begin{aligned} \mathbb{E}[|I_{t+1}|] &= \mathbb{E}[|I_t|] + 5\gamma/2 - (\gamma/(2\lambda))\mathbb{E}[\Pr(\text{err}_t = 1 | I_t, f_t, D_t)^2] + \mathbb{E}[\Pr(\text{inc}_t = 1 | I_t, f_t, D_t)] \\ &\leq \mathbb{E}[|I_t|] + 5\gamma/2 - (\gamma/(2\lambda))(\Pr(\text{err}_t = 1))^2 + \Pr(\text{inc}_t = 1), \end{aligned}$$

where the last step follows from the convexity of x^2 . Summing over all time steps gives us

$$\sum_{t=1}^T \mathbb{E}[|I_{t+1}|] \leq \sum_{t=1}^T \mathbb{E}[|I_t|] + 5\gamma T/2 - (\gamma/(2\lambda)) \sum_{t=1}^T (\Pr(\text{err}_t = 1))^2 + \sum_{t=1}^T \Pr(\text{inc}_t = 1).$$

Noting that $\mathbb{E}[|I_1|] = 1$ and $\mathbb{E}[|I_{T+1}|] \geq 0$, multiplying both sides by $2\lambda/(\gamma T)$, and rearranging terms gives us

$$\frac{1}{T} \sum_{t=1}^T (\Pr(\text{err}_t = 1))^2 \leq \frac{2\lambda}{\gamma T} + 5\lambda + \frac{2\lambda}{\gamma T} \sum_{t=1}^T \Pr(\text{inc}_t = 1) \leq \frac{2\lambda}{\gamma T}(K + 1) + 5\lambda. \quad (1)$$

The last inequality holds because $\sum_{t=1}^T \Pr(\text{inc}_t = 1) = \sum_{t=1}^T \mathbb{E}[\text{inc}_t] = \mathbb{E}[\sum_{t=1}^T \text{inc}_t] \leq K$. Applying Jensen's inequality to the left-hand side and taking the square root of both sides, we get

$$\frac{1}{T} \sum_{t=1}^T \Pr(\text{err}_t = 1) \leq \sqrt{\frac{2\lambda(K + 1)}{T\gamma} + 5\lambda} \leq \sqrt{\frac{2\lambda(K + 1)}{T\gamma}} + \sqrt{5\lambda}.$$

This allows us to bound the expected average error with

$$\mathbb{E}\left[\frac{1}{T} \sum_{t=1}^T (\text{err}_t + \text{inc}_t)\right] = \frac{1}{T} \sum_{t=1}^T \Pr(\text{err}_t = 1) + \frac{1}{T} \sum_{t=1}^T \Pr(\text{inc}_t = 1) \leq \sqrt{\frac{2\lambda(K + 1)}{T\gamma}} + \sqrt{5\lambda} + \frac{K}{T}.$$

■

The following theorem shows that the dependence on λ cannot be significantly improved. The proof is in the appendix.

Theorem 6 *Any algorithm for learning one-dimensional thresholds in the realizable setting (i.e., $K = 0$) under the definition of **near** stated above must suffer error $\Omega(\sqrt{\lambda})$.*

5.2 Hyperplanes

We now move on to the more interesting problem of efficiently learning hyperplanes with drift. For any two normalized vectors u and u' , let $\theta(u, u') = \arccos(u \cdot u')$ denote the angle between u and u' . We define $\text{near}(u, u')$ to hold if and only if $\theta(u, u') \leq \gamma$ for some fixed parameter $\gamma \in (0, \pi/2)$.

At each time step t , the adversary selects an arbitrary λ -good distribution D_t over unit-length d -dimensional points and a unit-length weight vector u_t such that the set u_1, \dots, u_T forms a K -shift legal sequence.³ The input point x_t is then drawn from D_t and assigned the label $\text{sign}(u_t \cdot x_t)$.

We analyze the Modified Perceptron algorithm originally proposed by Blum et al. [4] and later studied by Dasgupta et al. [7] in the context of active learning. This algorithm maintains a current weight vector w_t . The initial weight vector w_1 can be selected arbitrarily. At each time step t , when the algorithm observes the point x_t , it predicts the label $\text{sign}(w_t \cdot x_t)$. If the algorithm makes a mistake at time t , it sets $w_{t+1} = w_t - 2(w_t \cdot x_t)x_t$, otherwise no update is made and $w_{t+1} = w_t$. The factor of 2 in the update rule enforces that $\|w_t\| = 1$ for all t as long as $\|x_t\| = 1$. Note that unlike the algorithm for thresholds, this algorithm *does not* require any knowledge of γ .

We begin with a lemma which extends Lemma 3 of Dasgupta et al. [7] from a uniform distribution to a λ -good distribution. The intuition behind the proofs is similar. At a high level, we need to show that the adversary cannot place too much weight on points close to the algorithm's current decision boundary. Thus if the algorithm's probability of making a mistake is high, then there is a significant probability that the mistake will be on a point far from the boundary and significant progress will be made. In this lemma and the results that follow, let err_t be a random variable that is 1 if the algorithm makes an error at time t and 0 otherwise.

Lemma 7 *Consider the Modified Perceptron. At every time t , $w_{t+1} \cdot u_t \geq w_t \cdot u_t$. Furthermore, there exists a positive constant $c \leq 10$ such that for all t , for any $\eta \in (0, 1/2)$, if $\Pr(\text{err}_t | w_t, u_t, D_t) \geq 2c\eta\lambda/\gamma + 4\lambda$, then with probability at least $\Pr(\text{err}_t | w_t, u_t, D_t) - (2c\eta\lambda/\gamma + 4\lambda)$, we have $1 - w_{t+1} \cdot u_t \leq (1 - \eta^2/d)(1 - w_t \cdot u_t)$.*

The proof relies on the following fact about λ -good distributions under the current definition of **near**.

Lemma 8 *There exists a positive constant $c \leq 10$ such that for any $\eta \in (0, 1/2)$, for any d -dimensional vector w such that $\|w\| = 1$, for any λ -good distribution D , $\Pr_{x \sim D}(|w \cdot x| \leq \eta/\sqrt{d}) \leq c\eta\lambda/\gamma + 2\lambda$.*

The proof of this lemma is based on the following intuition. Consider the pair of hyperplanes corresponding to any two weight vectors w_1 and w_2 such that the angle between w_1 and w_2 is at most γ . Let Δ be the set of points x on which these hyperplanes disagree, i.e., all x such that $\text{sign}(w_1 \cdot x) \neq \text{sign}(w_2 \cdot x)$. Since D is λ -good, $D(\Delta) \leq \lambda$. The idea of the proof is to cover at least half of the points x such that $|w \cdot x| \leq \eta/\sqrt{d}$ using k sets like Δ , implying that the total weight D assigns to these points is at most $k\lambda$. In particular, we show that it is possible to construct such a cover with $k \leq 5\eta/\gamma + 1$, implying that the total probability D can place on points x such that $|w \cdot x| \leq \eta/\sqrt{d}$ is bounded by $10\eta\lambda/\gamma + 2\lambda$. The full proof appears in the appendix.

We are now ready to prove Lemma 7.

Proof of Lemma 7: The first half of the lemma is trivial if no mistake is made since $w_{t+1} = w_t$ in this case. If a mistake is made, then $w_{t+1} \cdot u_t = w_t \cdot u_t - 2(w_t \cdot x_t)(x_t \cdot u_t)$. Since there was an error, $\text{sign}(w_t \cdot x_t) \neq \text{sign}(x_t \cdot u_t)$ and $2(w_t \cdot x_t)(x_t \cdot u_t) < 0$.

For the second half, by Lemma 8 and the union bound, $\Pr(|w_t \cdot x_t| |u_t \cdot x_t| \leq \eta^2/d) \leq 2c\eta\lambda/\gamma + 4\lambda$. Thus if $\Pr(\text{err}_t | w_t, u_t, D_t) > 2c\eta\lambda/\gamma + 4\lambda$, then the probability that an error is made and $|w_t \cdot x_t| |u_t \cdot x_t| > \eta^2/d$ is at least $\Pr(\text{err}_t | w_t, u_t, D_t) - (2c\eta\lambda/\gamma + 4\lambda)$. Suppose this is the case. Then, as desired,

$$\begin{aligned} 1 - w_{t+1} \cdot u_t &= 1 - w_t \cdot u_t + 2(w_t \cdot x_t)(x_t \cdot u_t) = 1 - w_t \cdot u_t - 2|w_t \cdot x_t||x_t \cdot u_t| \\ &\leq 1 - w_t \cdot u_t - \frac{2\eta^2}{d} \leq 1 - w_t \cdot u_t - \frac{2\eta^2}{d} \frac{1 - w_t \cdot u_t}{2} = (1 - w_t \cdot u_t) \left(1 - \frac{\eta^2}{d}\right). \end{aligned}$$

■

Corollary 9 below follows from a simple application of technical properties of the cosine function.

Corollary 9 *There exists a constant $c \leq 10$ such that for any $\eta \in (0, 1/2)$, if $\Pr(\text{err}_t | w_t, u_t, D_t) > 2c\eta\lambda/\gamma + 4\lambda$, then with probability at least $\Pr(\text{err}_t | w_t, u_t, D_t) - (2c\eta\lambda/\gamma + 4\lambda)$,*

$$\theta(w_{t+1}, u_t) \leq \sqrt{1 - \frac{\eta^2}{d}} \theta(w_t, u_t) \leq \left(1 - \frac{\eta^2}{2d}\right) \theta(w_t, u_t).$$

³The assumption that $\|u_t\| = \|x_t\| = 1$ simplifies our presentation of results and nothing more. By modifying the definition of **near** and the update rule in a straight-forward manner, all of the results in this section can be extended to hold when the assumption is not true.

Finally, the following theorem uses these lemmas to bound the average error of the Modified Perceptron. The evolution of the angle between w_t and u_t is analyzed over time, similarly to how the evolution of $|I_t|$ was analyzed in the proof of Theorem 5. The result for general values of K is obtained by breaking the sequence down into (no more than) $K + 1$ subsequences on which there is no shift, applying a similar analysis on each subsequence, and summing the error bounds. This analysis is possible only if the time steps at which the shifts occur are fixed in advance, though it does *not* require that the algorithm is aware of the shifts in advance. The optimal setting of η and a brief interpretation of the resulting bounds are given below.

Theorem 10 *Let \mathcal{F} be the class of hyperplanes and let $\mathbf{near}(u, u')$ hold if and only if $\arccos(u \cdot u') \leq \gamma$ for a fixed parameter $\gamma \leq \lambda$. There exists a constant $c \leq 10$ such that when $K = 0$, for any $\eta \in (0, 1/2)$, the expected average error of the Modified Perceptron algorithm is no more than*

$$\left(1 + \frac{2\pi}{T\gamma} + \frac{\eta^2}{2d}\right) \frac{\lambda d}{q\eta^2} + 2q$$

where $q = (c\eta\lambda/\gamma + 2\lambda)$.

If the adversary chooses the time steps t at which a shift will occur in advance (yet, unknown to the learner), then for any K , for any $\eta \in (0, 1/2)$, the expected average error of the Modified Perceptron algorithm is no more than

$$\frac{K+1}{T} + \left(1 + \frac{2\pi(K+1)}{T\gamma} + \frac{\eta^2}{2d}\right) \frac{\lambda d}{q\eta^2} + 2q.$$

The bounds stated in this theorem can be difficult to interpret. Before jumping into the proof, let us take a moment to examine them in more detail to understand what this theorem really means. Setting $\eta = (d/\lambda)^{1/4}\gamma^{1/2}$ in Theorem 10, we obtain that when $T \gg (K+1)/\gamma$, the average error is bounded by $O(\lambda^{1/4}d^{1/4}\sqrt{\lambda/\gamma})$. If we think of γ as a constant fraction of λ , then this bound is essentially $O(\lambda^{1/4}d^{1/4})$. We do not know if it is possible to improve this bound to achieve an error of $O(\sqrt{\lambda d})$, which would match the bound of the inefficient algorithm presented in Section 4. Certainly such a bound would be desirable. However, this bound tells us that for hyperplanes, some amount of drift-resistance is possible with an efficient algorithm.

Note that in order for the bound to be non-trivial, γ must be small compared to $1/d$, in which case η is less than $1/2$.

Proof of Theorem 10: We first prove the result for $K = 0$ and then briefly discuss the how to extend the proof to cover general values of K .

Let $\theta_t = \theta(w_t, u_t)$. By definition, $\Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t) \leq (\theta_t/\gamma + 1)\lambda$, and $\theta_t \geq \Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t)\gamma/\lambda - \gamma$. By Lemma 7, for all t , $w_{t+1} \cdot w_t \geq w_t \cdot u_t$. Thus $\theta(w_{t+1}, u_t) \leq \theta(w_t, u_t)$, and since we have assumed no shifts, $\theta_{t+1} \leq \theta_t + \gamma$. We will show that this implies that for any t ,

$$\mathbb{E}[\theta_{t+1} | w_t, u_t, D_t] \leq \theta_t + \left(1 + \frac{\eta^2}{2d}\right) \gamma - (\Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t) - 2q) \frac{\eta^2 q \gamma}{d\lambda}, \quad (2)$$

where $q = (c\eta\lambda/\gamma + 2\lambda)$. This clearly holds if $\Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t) \leq 2q$ since η^2 and d are positive and in this case the last term is negative. Suppose instead that $\Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t) > 2q = 2(c\eta\lambda/\gamma + 4\lambda)$. By Corollary 9 and the bounds above,

$$\begin{aligned} \mathbb{E}[\theta_{t+1} | w_t, u_t, D_t] &\leq (\Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t) - 2q) \left(1 - \frac{\eta^2}{2d}\right) \theta_t + (1 - (\Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t) - 2q)) \theta_t + \gamma \\ &\leq \theta_t + \gamma - (\Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t) - 2q) \frac{\eta^2}{2d} \left(\frac{\Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t) \gamma}{\lambda} - \gamma\right) \\ &\leq \theta_t + \gamma - (\Pr(\mathbf{err}_t = 1 | w_t, u_t, D_t) - 2q) \frac{\eta^2}{2d} \left(\frac{2q\gamma}{\lambda} - \gamma\right) \end{aligned}$$

which implies Equation 2. Now, taking the expectation over $\{w_t, u_t, D_t\}$ of both sides of Equation 2, we get that for any t

$$\mathbb{E}[\theta_{t+1}] \leq \mathbb{E}[\theta_t] + \left(1 + \frac{\eta^2}{2d}\right) \gamma - (\Pr(\mathbf{err}_t) - 2q) \frac{\eta^2 q \gamma}{d\lambda}.$$

Summing over time steps, we then have that

$$\sum_{t=1}^T \mathbb{E}[\theta_{t+1}] \leq \sum_{t=1}^T \mathbb{E}[\theta_t] + \left(1 + \frac{\eta^2}{2d}\right) \gamma T - \frac{\eta^2 q \gamma}{d \lambda} \left(\sum_{t=1}^T \Pr(\mathbf{err}_t) - 2qT \right).$$

Since $\theta_t \in [0, 2\pi)$ for all t , this implies that

$$0 \leq 2\pi + \left(1 + \frac{\eta^2}{2d}\right) \gamma T - \frac{\eta^2 q \gamma}{d \lambda} \left(\sum_{t=1}^T \Pr(\mathbf{err}_t) - 2qT \right).$$

Rearranging terms and multiplying both sides by $d\lambda/(\eta^2 q \gamma)$ yields

$$\sum_{t=1}^T \Pr(\mathbf{err}_t) \leq \frac{2\pi d \lambda}{\eta^2 q \gamma} + \left(1 + \frac{\eta^2}{2d}\right) \frac{d \lambda}{\eta^2 q} T + 2qT.$$

Dividing both sides by T gives the desired bound on error.

To get the bound for general K , note that the analysis leading up to this past equation can be applied to all subsequences during which there is no shift. Summing the above bound for all such subsequences (where T is replaced by the length of the subsequence) with $\Pr(\mathbf{err}_t)$ pessimistically bounded by 1 whenever a shift occurs between times t and $t + 1$ leads to the bound. ■

6 Simulations

In this section, we discuss a series of simulations on synthetic data designed to illustrate the effectiveness of different algorithms for learning drifting hyperplanes. In particular, we compare the performance of the standard Perceptron algorithm [16], the Shift Perceptron [5], the Randomized Budget Perceptron [5], and the Modified Perceptron [4, 7] analyzed above. Like the Perceptron algorithm, the Shift Perceptron maintains a vector of weights, but each time a mistake is made, the Shift Perceptron shrinks its current weight vector towards zero in a way that depends on both the current number of mistakes and a parameter λ . The Randomized Budget Perceptron is similar but additionally tracks the set of examples that contribute to its current weight vector. If the size of this set exceeds a predetermined budget B , one example is randomly removed and the weight vector is updated by removing the contribution of this example.

For each experiment, the sequence $\mathbf{x}_1, \mathbf{x}_2, \dots$ of synthetic data points was generated as follows. We first generated 5000 d -dimensional random points $\mathbf{z}_1, \dots, \mathbf{z}_{5000}$ drawn from a zero-mean unit-covariance Gaussian distribution. We then generated a random $D \times d$ linear transformation matrix A , and used this to project each d -dimensional point \mathbf{z}_t to a D -dimensional vector $\mathbf{x}_t = A\mathbf{z}_t$. The resulting data points were thus D -dimensional points with a true underlying dimension of d . We fixed $D = 1000$ and experimented with various values of d between 5 and 500.

We generated each sequence of randomly drifting target weight vectors $\mathbf{u}_1, \mathbf{u}_2, \dots$ as follows. To start, \mathbf{u}_1 was drawn from a zero-mean unit-covariance D -dimensional Gaussian distribution. In the first set of experiments, which we refer to as the *random drift* experiments, each subsequent target \mathbf{u}_t was set to $\mathbf{u}_{t-1} + \delta_t$ where $\delta_t \sim \mathcal{N}(\mathbf{0}, \sigma I)$ for $\sigma = 0.1$. In the second set of experiments, which we refer to as the *linear drift* experiments, each \mathbf{u}_t was set to $\mathbf{u}_{t-1} + \delta$ for a fixed random vector δ . Each set of experiments was repeated 1000 times.

Both the Shift Perceptron and the Randomized Budget Perceptron are tuned using a single parameter (denoted by λ and B respectively). While we originally planned to tune these parameters using additional random draws of the data, the best values of these parameters simply reduced each algorithm to the original Perceptron. Instead, we set $\lambda = 0.01$ or $\lambda = 0.0001$, and $B = 300$, as these values resulted in fairly typical behavior for each of the algorithms.

The results are summarized in Figure 1. The two left plots show the results for the random drift experiments with $d = 5$. The two right plots show the results for the linear drift experiments with $d = 50$. The two top plots show the cumulative number of mistakes made by each of the four algorithms averaged over 1000 runs, while the bottom two plots show *difference* between the cumulative number of mistakes made by the Perceptron and the cumulative number of mistakes made by each algorithm. (Values above 0 indicate that an algorithm made more mistakes than the Perceptron, while values below 0 indicate than an algorithm made fewer.) The error bars correspond to the 95% confidence interval over the 1000 runs.

Consider the top-left plot summarizing the results of the random drift experiments. We see that all algorithms made between 250 and 300 mistakes, but the Modified Perceptron (green circles) made (statistically significantly) fewer mistakes than the others. This difference is easier to see in

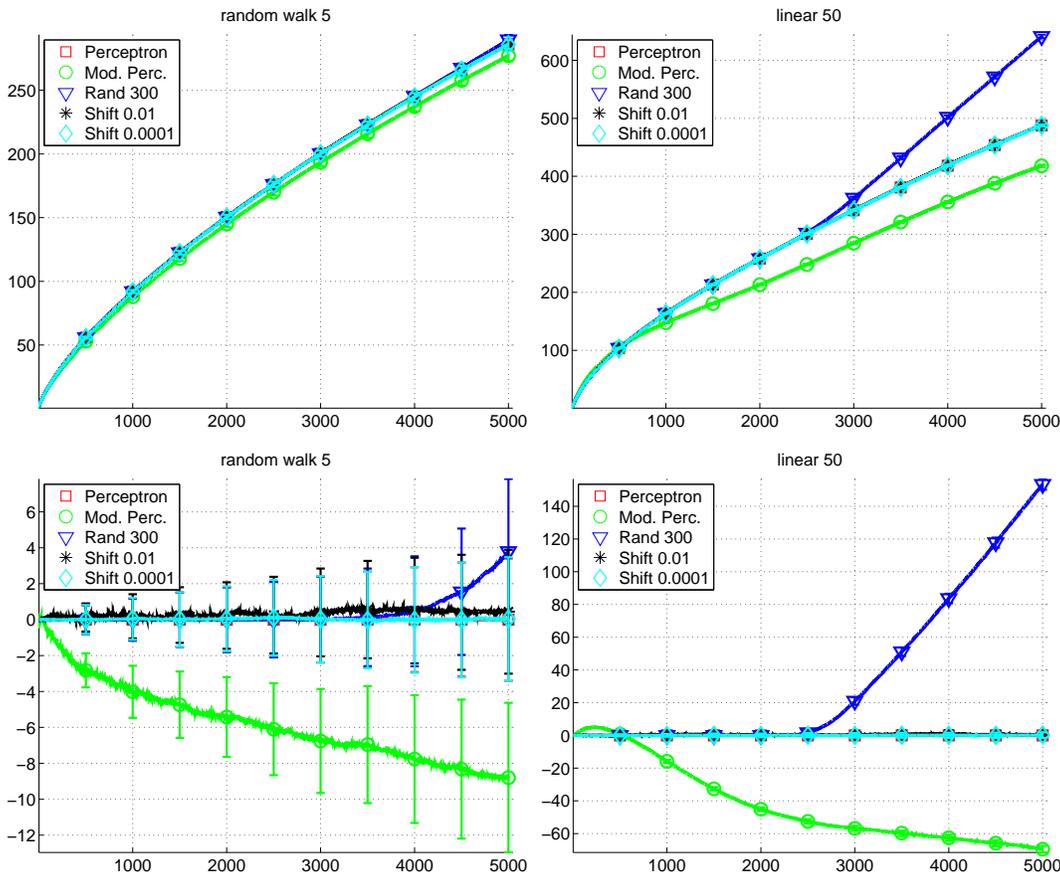


Figure 1: Cumulative number of mistakes (top) and difference between the cumulative number of mistakes and the cumulative number of mistakes made by the Perceptron algorithm (bottom) averaged over 1000 runs for the random drift experiments (left) and linear drift experiments (right). Four algorithms are evaluated: The standard Perceptron (red squares, largely hidden behind the Shift Perceptron), the Modified Perceptron (analyzed above, green circles), the Random Budget Perceptron with $B = 300$ (blue triangles), and the Shift Perceptron with $\lambda = 0.1$ (black stars) and $\lambda = 0.0001$ (teal diamonds). The bars indicate 95% confidence intervals.

the bottom-left plot. The Shift Perceptron made about 2% more mistakes than the Perceptron throughout the entire run. The Randomized Budget Perceptron was identical to the Perceptron algorithm until its budget of examples is exceeded, but overall made 1.2% more mistakes. Finally, during a prefix of training the Modified Perceptron made more mistakes than the Perceptron, but after about 500 training examples, the Modified Perceptron outperformed the Perceptron, making about 4% fewer mistakes.

The two right plots show qualitatively similar results for the linear drift experiments. During the first 500 examples the Modified Perceptron made more mistakes than the Perceptron algorithm, but it eventually performs better, ending with 15% fewer mistakes. As before, the performance of the Randomized Budget Perceptron started to degrade after its number of mistakes exceeded the budget. Finally, the Shift Perceptron made 4% more mistakes than the Perceptron after 1000 examples.

The total number of mistakes made by each of the four algorithms for various values of d are shown in the top two panels of Figure 2. As before, the bottom panels show the performance relative to the Perceptron, with values above 0 corresponding to more mistakes than the Perceptron. The two left plots show the results for the random drift experiments and the two right plots for the linear drift.

Comparing the top plots we observe that the random drift setting is slightly harder than the linear drift setting for lower values of d . For example, for $d = 20$ most algorithms made about

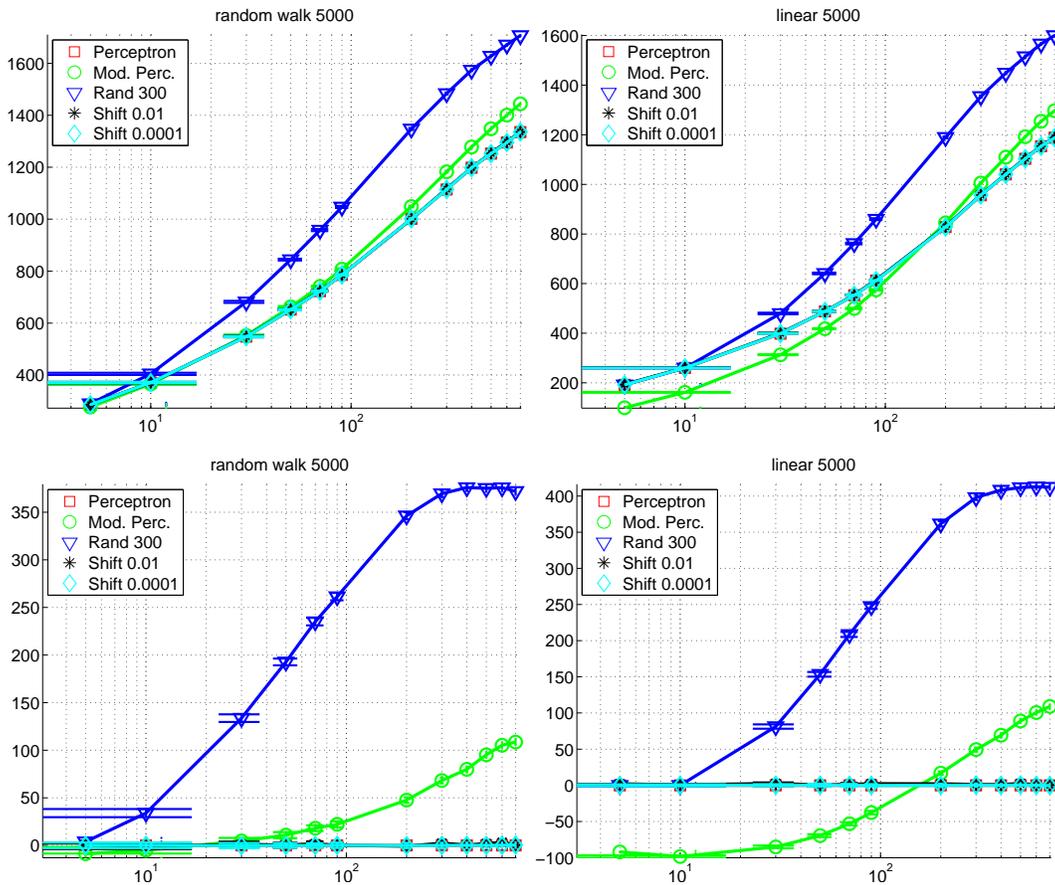


Figure 2: Total number of mistakes (top) and difference between the total number of mistakes and the number of mistakes made by the Perceptron algorithm (bottom) for different values of the underlying dimension d of the data for the random drift experiments (left) and the linear drift experiments (right). Again, the bars indicate 95% confidence intervals. (The elongation of these bars toward the edge of the plot is only an artifact of the log-scale axis.)

550 errors in the random drift setting but only 400 mistakes in the linear drift setting. For high values of d this gap is reduced. For small values of d , the Modified Perceptron outperformed the other algorithms, especially in the linear drift setting. For example, it made 100 fewer mistakes than the Perceptron algorithm with $d = 5$. When the underlying dimension d was large it made more mistakes compared to the other algorithms. The break-even point is about $d = 15$ for random drift and $d = 150$ for linear drift. The reason for this phenomenon is not clear to us. Note, however, that the underlying dimension d plays a major role in determining the difficulty of each problem, while the actual dimension D matters less. (This is not apparent from the experiments presented here, but we found it to be true when experimenting with different values of D .) This observation is in line with the dimension-independent bounds commonly published in the literature.

References

- [1] P. L. Bartlett. Learning with a slowly changing distribution. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992.
- [2] P. L. Bartlett, S. Ben-David, and S. Kulkarni. Learning changing concepts by exploiting the structure of change. *Machine Learning*, 41:153–174, 2000.
- [3] R. D. Barve and P. M. Long. On the complexity of learning from drifting distributions. *Information and Computation*, 138(2):101–123, 1997.

- [4] A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22:35–52, 1998.
- [5] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2/3):143–167, 2007.
- [6] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.
- [7] S. Dasgupta, A. Tauman Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *Proceedings of the 18th Annual Conference on Learning Theory*, 2005.
- [8] Y. Freund and Y. Mansour. Learning under persistent drift. In *Proceedings of EuroColt*, pages 109–118, 1997.
- [9] E. Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.
- [10] D. P. Helmbold and P. M. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14(1):27–46, 1994.
- [11] M. Herbster and M. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–78, 1998.
- [12] M. Herbster and M. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.
- [13] A. Kuh, T. Petsche, and R. L. Rivest. Learning time-varying concepts. In *Advances in Neural Information Processing Systems 3*, 1991.
- [14] A. Kuh, T. Petsche, and R. L. Rivest. Incrementally learning time-varying half-planes. In *Advances in Neural Information Processing Systems 4*, pages 920–927, 1992.
- [15] P. M. Long. The complexity of learning according to two models of a drifting environment. *Machine Learning*, 37:337–354, 1999.
- [16] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

A Additional Proofs

A.1 Proof of Theorem 6

We describe a strategy that the adversary can employ in order to force any learning algorithm to make a mistake with probability at least $(1 - 1/e)/2$ every $2/\sqrt{\lambda}$ time steps, resulting in an average error of $\Omega(\sqrt{\lambda})$.

The strategy for the adversary is simple. At time $t = 0$, the adversary sets f_1 such that $\tau_t = 1/2$. The adversary then chooses a random bit b which is 1 with probability $1/2$ and 0 with probability $1/2$. If $b = 1$, then the adversary gradually shifts the threshold to the right, increasing it by γ each time step until it reaches $1/2 + \gamma/\sqrt{\lambda}$, and then shifts it back again. On the other hand, if $b = 0$, then the adversary fixes $\tau_t = 1/2$ for $t = 1$ to $2/\sqrt{\lambda}$. In either case, at each of time step, the adversary sets D_t to be any λ -good distribution for which weight $\sqrt{\lambda}$ is spread uniformly over the region $[1/2, 1/2 + \gamma/\sqrt{\lambda}]$. After time $2/\sqrt{\lambda}$, the process repeats with a new random bit b .

Let us consider the probability that the learning algorithm makes at least one mistake during the first $2/\sqrt{\lambda}$ time steps. Because the learning algorithm does not know the random bit b , the algorithm cannot know whether the target is shifting or fixed, even if it is aware of the adversary’s strategy. Therefore, the first time that the algorithm sees a point x_t in $(1/2, 1/2 + t\gamma)$ for $t \in \{1, \dots, 1/\sqrt{\lambda}\}$ or a point x_t in $(1/2, 1/2 + (2/\sqrt{\lambda} - t)\gamma)$ for $t \in \{1/\sqrt{\lambda} + 1, \dots, 2/\sqrt{\lambda}\}$, it makes a mistake with probability $1/2$. The algorithm will see at least one such point with probability $1 - \prod_{t=1}^{1/\sqrt{\lambda}} (1 - t\lambda)^2 = 1 - e^{-\sum_{t=1}^{1/\sqrt{\lambda}} 2\ln(1-t\lambda)} \geq 1 - e^{-\sum_{t=1}^{1/\sqrt{\lambda}} -2t\lambda} \geq 1 - e^{-1}$, where the first inequality follows from the fact that

$\ln(x) \leq x - 1$ and the second from the fact that $\sum_{t=1}^{1/\sqrt{\lambda}} t \geq 1/(2\lambda)$. This implies that the probability that the algorithm makes at least one mistake during one of the first $2/\sqrt{\lambda}$ time steps is $(1 - 1/e)/2$.

The same analysis can be repeated to show that this is true of each consecutive interval of $2/\sqrt{\lambda}$ steps. Thus the error rate of any algorithm is at least $(1 - 1/e)\sqrt{\lambda}/4$.

A.2 Proof of Lemma 8

Let U be the uniform distribution over d -dimensional unit vectors. For any d -dimensional vector x such that $\|x\| = 1$, $\Pr_{z \sim U}(|z \cdot x| > 1/(2\sqrt{d})) \geq 1/2$ (see Dasgupta et al. [7]). Let I be an indicator function that is 1 if its input is true and 0 otherwise. For any distribution Q over d -dimensional unit vectors,

$$\begin{aligned} \sup_{z: \|z\|=1} \Pr_{x \sim Q}(|z \cdot x| > 1/(2\sqrt{d})) &\geq \mathbb{E}_{z \sim U} \left[\mathbb{E}_{x \sim Q} \left[I(|z \cdot x| > 1/(2\sqrt{d})) \right] \right] \\ &= \mathbb{E}_{x \sim Q} \left[\Pr_{z \sim U}(|z \cdot x| > 1/(2\sqrt{d})) \right] \geq 1/2. \end{aligned}$$

This implies that for any distribution Q there exists a vector z such that $\Pr_{x \sim Q}(|z \cdot x| > 1/(2\sqrt{d})) \geq 1/2$. For the remainder of the proof we let Q be the distribution D conditioned on $|w \cdot x| \leq \eta/\sqrt{d}$, and define z to be any vector satisfying the property above for Q .

Let $w^+ = w + 2\eta z$ and $w^- = w - 2\eta z$. Let X be the set of all unit vectors x such that $|z \cdot x| > 1/(2\sqrt{d})$ and $|w \cdot x| \leq \eta/\sqrt{d}$. It is easy to verify that for all $x \in X$, $\text{sign}(w^+ \cdot x) \neq \text{sign}(w^- \cdot x)$. Furthermore, we can construct a sequence of unit vectors w_0, \dots, w_k such that $w_0 = w^+/\|w^+\|$ and $w_k = w^-/\|w^-\|$, $\arccos(w_i \cdot w_{i+1}) < \gamma$, and $k \leq 5\eta/\gamma + 1$. To see how, let θ be the angle between w_0 and w_k . Then $\cos(\theta) = (w^+ \cdot w^-)/(\|w^+\| \|w^-\|) \geq (1 - 4\eta^2)/(1 + 4\eta^2) > 1 - 8\eta^2$, where the first inequality follows from the fact that $\|w^+\| \|w^-\| = \sqrt{(1 + 4\eta^2 + 4\eta(w \cdot z))(1 + 4\eta^2 - 4\eta(w \cdot z))} = \sqrt{(1 + 4\eta^2)^2 - 16\eta^2(w \cdot z)^2} \leq 1 + 4\eta^2$.

Since $\eta < 1/2$ we have that $\theta \in [0, \pi/2]$. It can be shown that for any $\theta \in [0, \pi/2]$, $(4/\pi^2)\theta^2 \leq 1 - \cos(\theta)$. This implies that $\theta < \sqrt{2}\pi\eta < 5\eta$. Since the angle between each pair w_i and w_{i+1} can be γ , we can create a sequence of vectors satisfying the property above with $k \leq \lceil 5\eta/\gamma \rceil \leq 5\eta/\gamma + 1$.

We have established that for every $x \in X$, $\text{sign}(w^+ \cdot x) \neq \text{sign}(w^- \cdot x)$. This implies that for every x there is an i such that $\text{sign}(w_i \cdot x) \neq \text{sign}(w_{i+1} \cdot x)$. Thus to bound the weight of X under D , it suffices to bound the weight of the regions $\Delta_i = \{x : \text{sign}(w_i \cdot x) \neq \text{sign}(w_{i+1} \cdot x)\}$. Since, by construction, the angle between each adjacent pair of vectors is at most γ and D is λ -good, D places weight no more than λ on each set Δ_i , and no more than $k\lambda \leq 5\eta\lambda/\gamma + \lambda$ on the set X .

Finally, since we have shown that $\Pr_{x \sim Q}(|z \cdot x| > 1/(2\sqrt{d})) \geq 1/2$, it follows that $\Pr_{x \sim D}(|w \cdot x| \leq \eta/\sqrt{d}) \leq 2D(X) \leq 10\eta\lambda/\gamma + 2\lambda$. \blacksquare

Strongly Non-U-Shaped Learning Results by General Techniques

John Case

Department of Computer and Information Sciences,
University of Delaware, Newark, DE 19716-2586, USA.
case@cis.udel.edu

Timo Kötzing*

Max-Planck Institute for Informatics, 66123 Saarbrücken, Germany
koetzing@mpi-inf.mpg.de

Abstract

In learning, a semantic or behavioral U-shape occurs when a learner first learns, then unlearns, and, finally, relearns, some target concept (on the way to success). Within the framework of Inductive Inference, previous results have shown, for example, that such U-shapes are unnecessary for explanatory learning, but are necessary for behaviorally correct and non-trivial vacillatory learning. Herein we focus more on syntactic U-shapes.

This paper introduces two general techniques and applies them especially to syntactic U-shapes in learning: one technique to show when they are necessary and one to show when they are unnecessary. The technique for the former is very general and applicable to a much wider range of learning criteria. It employs so-called *self-learning classes of languages* which are shown to *characterize* completely one criterion learning more than another.

We apply these techniques to show that, for set-driven and partially set-driven learning, any kind of U-shapes are unnecessary. Furthermore, we show that U-shapes are *not* unnecessary in a strong way for iterative learning, contrasting an earlier result by Case and Moelius that semantic U-shapes *are* unnecessary for iterative learning.

1 Introduction

In Section 1.1 we explain U-shaped learning. In Section 1.2 we briefly discuss the general techniques of the present paper and summarize in Section 1.3 our applications of these techniques regarding the necessity of U-shaped learning.

1.1 U-Shaped Learning

U-shaped learning occurs when a learner first learns a correct *behavior*, then abandons that correct behavior and finally returns to it once again. This pattern of learning has been observed by cognitive and developmental psychologists in a variety of child development phenomena, such as language learning [SS82], understanding of temperature [SS82], weight conservation [SS82], object permanence [SS82] and face recognition [Car82]. The case of language acquisition is paradigmatic. For example, a child first uses *spoke*, the correct past tense of the irregular verb *speak*. Then the child overregularizes incorrectly using *speaked*. Lastly the child returns to using *spoke*. The language acquisition case of U-shaped learning behavior has figured prominently in cognitive science [MPU⁺92, TA02].

While the prior cognitive science literature on U-shaped learning was typically concerned with modeling *how* humans achieve U-shaped behavior, [BCM⁺08, CCJS07] are motivated by the question of *why* humans exhibit this seemingly inefficient behavior. Is it a mere harmless evolutionary inefficiency or is it *necessary* for full human learning power? A technically answerable version of this question is: are there some formal learning tasks for which U-shaped behavior is logically necessary? We first need to describe some formal criteria of successful learning.

An algorithmic learning function h is, in effect, fed an infinite sequence consisting of the elements of a (formal) language L in arbitrary order with possibly some pause symbols $\#$ in between elements.

*Timo Kötzing was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant no. NE 1182/5-1.

During this process h outputs a corresponding sequence $p(0), p(1), \dots$ of hypotheses (grammars) which may generate the language L to be learned. A fundamental criterion of successful learning of a language is called *explanatory learning* (TxtEx-learning) and was introduced by Gold [Gol67]. Explanatory learning requires that the learner’s output conjectures stabilize in the limit to a *single* conjecture (grammar/program, description/explanation) that generates the input language. *Behaviorally correct learning* [CL82, OW82] requires, for successful learning, only convergence in the limit to possibly infinitely many syntactically distinct but correct conjectures. Another interesting class of criteria features *vacillatory learning* [Cas99, JORS99]. This paradigm involves learning criteria which allow the learner to vacillate in the limit between *at most* some bounded, finite number of syntactically distinct but correct conjectures. For each criterion that we consider above (and below), a *non-U-shaped learner* is naturally modeled as a learner that never *semantically* returns to a previously abandoned correct conjecture on languages it learns according to that criterion.

[BCM⁺08] showed that every TxtEx-learnable class of languages is TxtEx-learnable by a non-U-shaped learner, that is, for TxtEx-learnability, U-shaped learning is *not* necessary. Furthermore, based on a proof in [FJO94], [BCM⁺08] noted that, by contrast, for behaviorally correct learning [CL82, OW82], U-shaped learning *is* necessary for full learning power. In [CCJS07] it is shown that, for non-trivial vacillatory learning, U-shaped learning is again necessary (for full learning power). Thus, in many contexts, seemingly inefficient U-shaped learning can actually increase one’s learning power.

What turns out to be a variant of non-U-shaped learning is *strongly non-U-shaped* learning essentially defined in [Wie91],¹ where the learner is required never to *syntactically* abandon a correct conjecture on languages it learns according to that criterion. Clearly, *strong* non-U-shaped learnability implies non-U-shaped learnability.² In our experience, for theoretical purposes, it is frequently easier to show non-U-shaped learnability by showing *strong* non-U-shaped learnability. Herein we especially study strong non-U-shaped learnability.

1.2 Presented Techniques

The present paper presents two general techniques to tackle problems regarding U-shaped learning.

The first general technique can be used to show the *necessity* of U-shapes and employs so-called *self-learning classes of languages*. These are explained in Section 3 below. These self-learning classes of languages provide a general way for finding classes of languages that separate two learning criteria, i.e., they give a general way of finding an *example* class of languages learnable with a given learning criterion, but not with another. Theorem 3.6 implies that its presented self-learning classes *necessarily* separate two learnability sets — *if any class does*. This technique is not specialized only to analyze U-shaped learning, but can be applied to other learning criteria as well. The technique is developed and discussed further in Section 3.

The second general technique is used to show that U-shapes are *unnecessary* and is phrased in terms of sufficient conditions for the non-U-shaped learnability of classes of languages. Section 4’s Theorems 4.8 and 4.9 provide these sufficient conditions, each for a different kind of U-shapes. Theorem 4.8 actually *characterizes* strong non-U-shaped learnability.

1.3 Applications of General Techniques

A learning machine is *set-driven* [WC80, SR84, Ful90, JORS99] (respectively, *partially set-driven* [SR84, Ful90, JORS99]) iff, at any time, its output conjecture depends only on the *set* of numerical data it has seen (respectively, set and data-sequence length), not on the order of that data’s presentation.³ Child language learning may be insensitive to the order or timing of data presentation; set-drivenness and partial set-drivenness provide two *local* notions of such insensitivity [Cas99]. It is interesting, then, to see the interaction of these notions with forbidding U-shapes of one kind or another. As we shall see in Section 5, Theorems 5.1 and 5.2, proved with the aid of a general technique from Section 4, imply, for these local data order insensitivity notions, for TxtEx-learning, U-shapes, *even in the strong sense* are unnecessary.

¹Wiehagen actually used the term *semantically finite* in place of *strongly non-U-shaped*. However, there is a clear connection between this notion and that of *non-U-shapedness*. Our choice of terminology is meant to expose this connection. See also [CM08a].

²For non-U-shaped learning, the learner (on the way to success) must not *semantically* abandon a correct conjecture. In general, semantic change of conjecture is not algorithmically detectable, but syntactic change is. However, in the cognitive science lab we can many times see a *behavioral/semantic* change, but it is beyond the current state of the art to see, for example, grammars in people’s heads — so we can’t *yet* see mere syntactic changes in people’s heads.

³Note that partially set-driven learning is also known as *rearrangement independent learning*.

An *iterative* learner outputs its conjectures only on the basis of its immediately prior conjecture (if any) and its current datum. As we shall see in Section 5, iterative learning provides a (first) example of a setting in which non-U-shaped and strongly non-U-shaped learning are extensionally distinct: [CM08b] shows semantic U-shapes to be *unnecessary* in iterative learning, while Theorem 5.4 in the present paper implies that they are not *strongly* unnecessary. To prove this latter result, we actually employ a self-learning class of languages that is a bit easier to work with than the relevant one from Theorem 3.6 — although the latter *must* work too (by Theorems 3.6 and 5.4).⁴

Some of our proofs involve subtle infinitary program self-reference arguments employing (variants of) the Operator Recursion Theorem (ORT) from [Cas74, Cas94, JORS99].

1.4 Open Problems

Some problems regarding the necessity of U-shapes of one kind or another still remain open. An *iterative with counter learner* is an iterative learner which, in making its conjectures, also has access to the data-sequence length of the set of numerical data so far. For example, it is still open whether semantic U-shapes are necessary for iterative with counter learning — as asked in [CM08b]. If so, then the relevant self-learning class from Theorem 3.6 below *must* provide a separation.

See the end of Section 5 for some more open problems regarding the necessity of any one of the two kinds of U-shapes for learning criteria of the present paper.

2 Mathematical Preliminaries

Unintroduced computability-theoretic notions follow [Rog67].

Strings herein are finite and over the alphabet $\{0, 1\}$. $\{0, 1\}^*$ denotes the set of all such strings; ε denotes the empty string.

\mathbb{N} denotes the set of natural numbers, $\{0, 1, 2, \dots\}$. We do not distinguish between natural numbers and their *dyadic* representations as strings.⁵

We fix the 1-1 and onto pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ from [RC94, Section 2.3]. In particular, for all x, y ,

$$\langle x, y \rangle = \sum_{k=0}^m x_k 2^{2k+1} + \sum_{k=0}^n y_k 2^{2k}, \quad (1)$$

where $x = \sum_{k=0}^m x_k 2^k$, $y = \sum_{k=0}^n y_k 2^k$ and $x_0, \dots, x_m, y_0, \dots, y_n \in \{0, 1\}$. The *binary* representation of $\langle x, y \rangle$ is an interleaving of the *binary* representations of x and y , where we alternate x 's and y 's digits and start on the right with the least most significant y digit. For example, $\langle 15, 2 \rangle = 94$, since $15 = 1111$ (binary), $2 = 0010$ (binary), and $94 = 10101110$ (binary). Define π_1 and π_2 to be the functions such that, for all x and y ,

$$\pi_1(\langle x, y \rangle) = x; \quad (2)$$

$$\pi_2(\langle x, y \rangle) = y. \quad (3)$$

π_1 and π_2 are, respectively, called the first and second projection functions.

The symbols $\subseteq, \subset, \supseteq, \supset$ respectively denote the subset, proper subset, superset and proper superset relation between sets.

For sets A, B , we let $A \setminus B = \{a \in A \mid a \notin B\}$.

The quantifier $\forall^\infty x$ means “for all but finitely many x ”, the quantifier $\exists^\infty x$ means “for infinitely many x ”. For any set A , $\text{card}(A)$ denotes the cardinality of A .

\mathfrak{P} and \mathfrak{R} denote, respectively, the set of all partial and of all total functions $\mathbb{N} \rightarrow \mathbb{N}$. With dom and range we denote, respectively, domain and range of a given function.

We sometimes denote a partial function f of $n > 0$ arguments x_1, \dots, x_n in lambda notation (as in Lisp) as $\lambda x_1, \dots, x_n. f(x_1, \dots, x_n)$. For example, with $c \in \mathbb{N}$, $\lambda x. c$ is the constantly c function of one argument.

Whenever we consider tuples of natural numbers as input to $f \in \mathfrak{P}$, it is understood that the general coding function $\langle \cdot, \cdot \rangle$ is used to (left-associatively) code the tuples into a single natural number.

If $f \in \mathfrak{P}$ is not defined for some argument x , then we denote this fact by $f(x)\uparrow$, and we say that f on x *diverges*; the opposite is denoted by $f(x)\downarrow$, and we say that f on x *converges*. If f on x converges to p , then we denote this fact by $f(x)\downarrow = p$.

⁴Some recent papers [CK08, CK10] have also employed (different) self-learning classes for separations.

⁵The *dyadic* representation of a natural number x = the x -th finite string over $\{0, 1\}$ in *length-lexicographical order*, where the counting of strings starts with zero [RC94]. Hence, unlike with binary representation, lead zeros matter.

We say that $f \in \mathfrak{P}$ converges to p iff $\exists x_0 : \forall x \geq x_0 : f(x) \downarrow \in \{?, p\}$; we write $f \rightarrow p$ to denote this.⁶

Computability Notions

We let \mathcal{P} and \mathcal{R} , respectively, denote the set of all partial computable and all total computable functions from \mathbb{N} to \mathbb{N} ; let $\varphi_0, \varphi_1, \dots$ be any acceptable programming system (numbering) of \mathcal{P} [Rog67]. φ_p is the element of \mathcal{P} computed by the program in this system with numerical name p . Via numerical naming all general purpose real world programming systems are acceptable.

A set $L \subseteq \mathbb{N}$ is *computably enumerable (ce)* iff it is the domain of a partial computable function. Let \mathcal{E} denote the set of all ce sets. We let W be the mapping such that $\forall e : W(e) = \text{dom}(\varphi_e)$. For each e , we write W_e instead of $W(e)$. W is, then, a mapping from \mathbb{N} onto \mathcal{E} . We say that e is an index, or program, (in W) for W_e . These programs e constitute a hypothesis space for learning in the present paper.

In this paper, a *computable operator* is a mapping from any one (respectively, two) partial function(s) $\mathbb{N} \rightarrow \mathbb{N}$ into one such partial function such that there exists an algorithm which, when fed *any* enumeration(s) of the graph(s) of the input(s), it outputs *some* enumeration of the graph of the output. Rogers [Rog67] extensively treats the one-ary case of these operators and calls them *recursive operators*.

A *finite sequence* is a mapping with a finite initial segment of \mathbb{N} as domain (and range, $\subseteq \mathbb{N}$). \emptyset denotes the empty sequence (and, also, the empty set). The set of all finite sequences is denoted by Seq . For any given set $A \subseteq \mathbb{N}$, the set of all finite sequences of elements in A is denoted by $\text{Seq}(A)$. For each finite sequence σ , we will denote the first element, if any, of that sequence by $\sigma(0)$, the second, if any, by $\sigma(1)$ and so on. $\#\text{elems}(\sigma)$ denotes the number of elements in a finite sequence σ , that is, the cardinality of its domain.

Following [LV08], we define for all $x \in \mathbb{N}$: $\bar{x} = 1^{\text{size}(x)}0x$. Using this notation we can define a function $\langle \cdot \rangle_{\text{Seq}}$ coding arbitrarily long finite sequences of natural numbers into \mathbb{N} (represented dyadically) such that

$$\langle \sigma \rangle_{\text{Seq}} = \overline{\sigma(0)} \dots \overline{\sigma(\#\text{elems}(\sigma) - 1)}. \quad (4)$$

In particular, $\langle \emptyset \rangle_{\text{Seq}} = \varepsilon$.

For example the finite sequence $(4, 7, 10)_{\text{decimal}} = (01,000,011)_{\text{dyadic}}$ is coded as 11001 1110000 1110011 (but without the spaces, which were added for ease of reading).⁷

We use \diamond (with infix notation) to denote concatenation on sequences. With a slight abuse of notation, for a sequence σ and a natural number x , we let $\sigma \diamond x$ denote the sequence that starts with the sequence σ and then ends with x .

For any finite sequence σ such that $\#\text{elems}(\sigma) > 0$, we let $\text{last}(\sigma)$ be the last element of σ and σ^- be σ with its last element deleted. By convention, we set $\emptyset^- = \emptyset$.

Obviously, $\langle \cdot \rangle_{\text{Seq}}$ is 1-1 [LV08].

Henceforth, we will many times identify a finite sequence σ with its code number $\langle \sigma \rangle_{\text{Seq}}$. However, when we employ expressions such as $\sigma(x)$, $\sigma = f$ and $\sigma \subset f$, we consider σ as a *sequence*, not as a number.

For a partial function $f \in \mathfrak{P}$ and $i \in \mathbb{N}$, if $\forall j < i : f(j) \downarrow$, then $f[i]$ is defined to be the finite sequence $f(0), \dots, f(i-1)$.

We fix the following 1-1 coding for all finite subsets of \mathbb{N} . For each non-empty finite set $D = \{x_0 < \dots < x_n\}$, $\langle x_0, \dots, x_n \rangle_{\text{Seq}}$ is the code for D and $\langle \rangle_{\text{Seq}}$ is the code for \emptyset .

Henceforth, we will many times identify a finite set D with its code number. However, when we employ expressions such as $x \in D$, $\text{card}(D)$, $\max(D)$ and $D \subset D'$, we consider D and D' as *sets*, not as numbers.

The symbol $\#$ is pronounced *pause* and is used to symbolize “no new input data” in a text. For each (possibly infinite) sequence q , let $\text{content}(q) = (\text{range}(q) \setminus \{\#\})$.

Later, we will type infinite sequences as being in \mathfrak{R} , *but*, technically, *texts* (for languages $\subseteq \mathbb{N}$) are *infinite sequences*, but they may contain pauses ($\#$ s) which are not natural numbers. Also, finite initial segments of texts are example finite sequences which can contain pauses. In our coding above of finite sequences, we code only sequences of natural numbers (and not pauses). To get around this, we will assume from now on that $\mathbb{N} \cup \{\#\}$ is efficiently coded 1-1 onto \mathbb{N} , say, by coding $\#$ as 0 and $n \in \mathbb{N}$ as $(n+1)$. In this way texts can be thought of as $\in \mathfrak{R}$ and finite initial segments of texts can, then, be coded as sequences of natural numbers. However, for texts T and for finite initial segments

⁶ $f(x)$ converges should not be confused with f converges to.

⁷ 1100111100001110011 is of course the dyadic representation of some number $\in \mathbb{N}$.

of such T , we will, whenever we need to talk about the value of $T(m)$, ignore the coding and work, for example, with whether the actual (not coded) values of $T(m) = \#$ or $n \in \mathbb{N}$. This ignoring of the coding will be useful from time to time.

From now on, by convention, f, g and h with or without decoration range over (partial) functions $\mathbb{N} \rightarrow \mathbb{N}$; x, y with or without decorations range over \mathbb{N} ; σ, τ with or without decorations range over finite sequences of natural numbers; D with or without decorations ranges over finite subsets of \mathbb{N} .

We will make use of a padded variant of the s-m-n Theorem [Rog67]. Intuitively, s-m-n permits algorithmic storage of arbitrary data (and, hence, programs) inside any program. The suitable padded variant of s-m-n we use herein states that there is a strictly monotonic increasing total computable function s such that

$$\forall a, b, c : \varphi_{s(a,b)}(c) = \varphi_a(b, c). \quad (5)$$

In (5), φ -program $s(a, b)$ is essentially φ -program a with datum b stored inside. We will also use a suitably padded version of Case's *Operator Recursion Theorem* (**ORT**), providing *infinitary* self (and other) reference [Cas74, Cas94, JORS99]. **ORT** itself states that, for all computable operators $\Theta : \mathfrak{P} \rightarrow \mathfrak{P}$,

$$\exists e \in \mathcal{R} \forall a, b : \varphi_{e(a)}(b) = \Theta(e)(a, b). \quad (6)$$

In the padded version we employ, the function e will also be strictly monotone increasing.

2.1 Computability-Theoretic Learning

In this section we formally define several criteria for computability-theoretic learning.

A *language* is a **ce** set $L \subseteq \mathbb{N}$. Any total function $T : \mathbb{N} \rightarrow \mathbb{N} \cup \{\#\}$ is called a *text*. For any given language L , a *text for L* is a text T such that $\text{content}(T) = L$. With $\mathbf{Txt}(L)$ we denote the set of all texts for L .

A *sequence generating operator* is a computable operator β taking as arguments a function h (the learner) and a text T and that outputs a function p . We call p the *learning sequence* of h given T .

Intuitively, β defines how a learner can interact with a given text to produce a sequence of conjectures.

We define the sequence generating operators **G**, **Psd**, **Sd**, **ItCtr** and **It** as follows. **G**, **Psd**, **Sd**, **ItCtr** and **It**, respectively, stand for Gold [Gol67], partially set-driven [SR84, Ful85, Ful90, JORS99], set-driven [WC80, JORS99] iterative with counter [CM08b] and iterative [WC80, Wie76], respectively. For all h, T, i ,

$$\begin{aligned} \mathbf{G}(h, T)(i) &= h(T[i]); \\ \mathbf{Psd}(h, T)(i) &= h(\text{content}(T[i]), i); \\ \mathbf{Sd}(h, T)(i) &= h(\text{content}(T[i])); \\ \mathbf{ItCtr}(h, T)(i) &= \begin{cases} ?, & \text{if } i = 0; \\ h(\mathbf{ItCtr}(h, T)(i-1), T(i-1), i-1), & \text{otherwise;} \end{cases} \\ \mathbf{It}(h, T)(i) &= \begin{cases} ?, & \text{if } i = 0; \\ h(\mathbf{It}(h, T)(i-1), T(i-1)), & \text{otherwise.} \end{cases} \end{aligned}$$

Successful learning might require the learner to observe certain restrictions, for example non-U-shapedness. These restrictions are formalized in our next definition.

A *learning restriction* is a predicate on a learner and a language, parameterized with a sequence generating operator. We write the parameter as a subscript and give the following examples.

- No restriction: The constantly true predicate of the appropriate type **T**.
- Total Learner: $\forall \beta, h, L : \mathcal{R}_\beta(h, L) \Leftrightarrow h \in \mathcal{R}$.
- Non-U-shaped: $\forall \beta, h, L : \mathbf{NU}_\beta(h, L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L), \forall i : (W_{\beta(h, T)(i)} = L \Rightarrow W_{\beta(h, T)(i+1)} = W_{\beta(h, T)(i)})]$.
- Strongly non-U-shaped: $\forall \beta, h, L : \mathbf{SNU}_\beta(h, L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L) \forall i : (W_{\beta(h, T)(i)} = L \Rightarrow \beta(h, T)(i+1) = \beta(h, T)(i))]$.

We combine any two learning restrictions by intersecting them, and we denote this combination by juxtaposition.

In order to motivate and define another group of learning restrictions, we now define the concept of a stabilizer sequence (called “stabilizer segment” in [Ful90]).

Let β be a sequence generating operator. Let L be a language and $h \in \mathcal{P}$ a learner. A sequence $\sigma \in \text{Seq}(L)$ is said to be a β -stabilizer sequence of h on L iff

$$(\forall T \in \mathbf{Txt}(L) | \sigma \subseteq T) \forall i \geq \#elets(\sigma) : \beta(h, T)(\#elets(\sigma)) = \beta(h, T)(i); \quad (7)$$

Intuitively, a stabilizer sequence σ of h on L is a sequence of elements from L such that h on any text extending σ will never make a change of conjecture after having seen σ .

It is well known that, if a learner h TxtEx -learns a language L , then there is a stabilizer sequence of h on L (see [JORS99]). However, texts don't necessarily contain such a sequence as an initial segment. Below, we define a learning restriction that requires a learner and a language to have stabilizer sequences as initial sequences of *all* texts for the language.

- Stabilizing: $\forall \beta, h, L$:

$$\mathbf{Stab}_\beta(h, L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L) \exists i_0 : T[i_0] \text{ is a } \beta\text{-stabilizer sequence of } h \text{ on } L].$$

Let β be sequence generating operator. For a learner h and a language L , we define a β -sink of h on L to be a conjecture e such that

$$\forall T \in \mathbf{Txt}(L) \forall i_0 : [\beta(h, T)(i_0) = e \Rightarrow (\forall i \geq i_0) (\beta(h, T)(i) = e)]. \quad (8)$$

Intuitively, a sink is a conjecture never abandoned on texts for L .

We specialize the concept of a stabilizer sequence to a *sink-stabilizer sequence*. This will be technically helpful for some of our results.

- Sink-stabilizing: $\forall \beta, h, L$:

$$\mathbf{Sink}_\beta(h, L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L) \exists e : (\beta(h, T) \rightarrow e \wedge e \text{ is a } \beta\text{-sink of } h \text{ on } L)].$$

Clearly, for all β, h, L , $\mathbf{Sink}_\beta(h, L)$ implies $\mathbf{Stab}_\beta(h, L)$. Sink-stabilizing is of interest, as we show a characterization theorem (Theorem 4.8 below) of strong non-U-shaped learning in terms of sink-stabilizing learning.

In order to obtain at least a sufficient condition for (not necessarily strongly) non-U-shaped learning, we weaken the concept of a sink as follows. For now, let $f \in \mathcal{P}$. An f -weak β -sink of h on L is a conjecture e such that

$$\forall T \in \mathbf{Txt}(L) \forall i_0 : [\beta(h, T)(i_0) = e \Rightarrow (\forall i \geq i_0) (f(\beta(h, T)(i), e) = 1)]. \quad (9)$$

We would like to employ for f above $f_0 = \lambda e, e'. W_e = W_{e'}$, in order to capture the notion of a “never semantically abandoned conjecture;” however, this function is not computable. Instead, we will use functions “approximating” f_0 : Let

$$\mathcal{F} = \{f \in \mathcal{R} \mid \forall e, e' : (f(e, e') = 1 \Rightarrow W_e = W_{e'})\}^8 \quad (10)$$

For all $f \in \mathcal{F}$, we define the following learning restriction.

- f -weak β -sink-stabilizing: $\forall \beta, h, L$:

$$\mathbf{Weaksink}_\beta^f(h, L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L) \exists e : (\beta(h, T) \rightarrow e \wedge e \text{ is an } f\text{-weak } \beta\text{-sink of } h \text{ on } L)].$$

We are now ready to give some formal definitions for successful learning.

Definition 2.1.

- For this paper, a *learning criterion* is a pair (α, β) such that α is a learning restriction and β a sequence generating operator. We also write $\alpha\mathbf{Txt}\beta\mathbf{Ex}$ to denote the learning criterion (α, β) .
- Let (α, β) be a learning criterion and h a learner. We say that h $\alpha\mathbf{Txt}\beta\mathbf{Ex}$ -learns a language L iff $\alpha_\beta(h, L)$ and, for all texts T for L , $\beta(h, T)$ is total and there is e with $\beta(h, T) \rightarrow e$ and $W_e = L$.
- We denote the class of all languages $\alpha\mathbf{Txt}\beta\mathbf{Ex}$ -learned by h with $\alpha\mathbf{Txt}\beta\mathbf{Ex}(h)$. Abusing notation, we use $\alpha\mathbf{Txt}\beta\mathbf{Ex}$ to denote the set of all classes of languages $\alpha\mathbf{Txt}\beta\mathbf{Ex}$ -learnable by some learner (as well as the learning criterion).

⁸Intuitively, all $f \in \mathcal{F}$ only output 1 if the inputs are semantically equivalent.

- We omit α if $\alpha = \mathbf{T}$.

We let

$$\text{The30} = \{(\alpha, \beta) \mid \alpha \in \{\mathbf{T}, \mathbf{NU}, \mathbf{SNU}, \mathbf{Sink}, \mathbf{Stab}\}, \beta \in \{\mathbf{G}, \mathbf{Psd}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{It}\}\} \cup \{\bigcup_{f \in \mathcal{F}} (\mathbf{Weaksink}^f, \beta) \mid \beta \in \{\mathbf{G}, \mathbf{Psd}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{It}\}\}. \quad (11)$$

This is a set of thirty *particular* learning criteria especially considered in this paper.

As noted above in Section 1.2, Theorem 3.6 below applies to learning criteria separations well beyond our concerns in the present paper with showing U-shapes do (or don't) make a difference in learning power. In particular we will see that Theorem 3.6 applies to *all* pairs of learning criteria from The30. Indirectly in Section 4 and directly in Section 5, though, we are mainly concerned with pairs of criteria (I_0, I_1) , with each criterion from The30, *where* $I_0 = \mathbf{SNU}I_1$ (or $I_0 = \mathbf{NU}I_1$).

Starred Learners

For a learner h , possibly learning with restricted access to past data, we write $h^*(\sigma)$ for what the current output of h is after being fed the sequence σ of data items.

In particular, for $h \in \mathcal{P}$ and σ a sequence, we have the following.

- If h is a set-driven learner:

$$h^*(\sigma) = h(\text{content}(\sigma)). \quad (12)$$

- If h is a partially set-driven learner:

$$h^*(\sigma) = h(\text{content}(\sigma), \#\text{elets}(\sigma)). \quad (13)$$

3 Self-Learning Classes of Languages for Separations

In this section we discuss a way of showing U-shapes to be *necessary*. Formally, this is done via showing that a learnability class *separates* from its non-U-shaped variant.

The approach described below is very general and can be applied to show separation results in many other areas of computability-theoretic learning in the limit as well.

The key idea is that of *self-learning classes of languages*. In the previous literature, *self-describing* classes of languages have been used.⁹ A particularly simple example class of self-describing languages, taken from [CL82, Theorem 1], is

$$\mathcal{L}_0 = \{L \text{ recursive} \mid L \neq \emptyset \wedge W_{\min L} = L\}. \quad (14)$$

Intuitively, each $L \in \mathcal{L}_0$ gives a complete *description* of itself, encoded (as a W -index) within only finitely many (in fact, one) of its elements. It is well-known, using standard computability theoretic arguments, that these kind of classes of languages are very big (for example, \mathcal{L}_0 contains a finite variant of any given ce language, which can easily be seen using Kleene's Recursion Theorem).

Many variants of self-describing classes of languages have been used for separation results within computability-theoretic learning (see, for example, [BB75, CL82, CS83, Cas99, JORS99]). Showing a separation with a complicated self-describing class of languages sometimes requires a non-trivial learner (see, for extreme examples, [CJLZ99]).

We now take the technique of self-describing one step further. A *self-learning* class of languages is such that each element of each language of the class provides instructions for what to compute and output as a new hypothesis. Thus, all a learner needs to do is to execute the instructions given by its latest datum. For example, an informal¹⁰ learner h_1 can be defined such that

$$\forall \sigma, x : h_1(\sigma \diamond x) = \varphi_x(\sigma \diamond x). \quad (15)$$

Intuitively, h_1 interprets the latest datum as a program in the φ -system and runs this program on all known data. Variants of this h_1 can be defined to obtain learners with special additional properties, such as totality or set-drivenness (see Theorem 3.6).

In practice, the general scheme is as follows. Suppose we want to show, for two learning criteria I_0 and I_1 , $I_1 \setminus I_0 \neq \emptyset$. Then, we define a simple learner, for example h_1 above, and let \mathcal{L}_1 be the class of all languages I_1 -learned by h_1 . All that would remain to be shown is that \mathcal{L}_1 is not I_0 -learnable, which can often be done using **ORT**.

⁹See [JORS99]. In there, the term "self-describing" was used on page 71 in the context of function learning and extended on page 97 to language learning.

¹⁰Note that we ignore the possible input of $x = \#$.

Below, in Theorem 3.6, we give a *very general* result regarding some self-learning classes of languages *guaranteed* to witness separations when they exist. In order to do so, we proceed next by making some formal definitions.

For a function $e \in \mathcal{P}$ and a language L , we let $e(L) = \{e(x) \mid x \in L\}$; for a class of languages \mathcal{L} , we let $e(\mathcal{L}) = \{e(L) \mid L \in \mathcal{L}\}$.

Let $\mathcal{P}_{c1-1} \subseteq \mathcal{P}$ (repectively, $\mathcal{R}_{c1-1} \subseteq \mathcal{R}$), denote the set of all 1-1 partial (respectively, total) computable functions with computable domain and range.

Definition 3.1. A learning criterion I is called *computably \mathcal{P}_{c1-1} -robust* iff there is a computable operator $\Theta : \mathfrak{P}^2 \rightarrow \mathfrak{P}$ such that

$$\forall h \in \mathcal{P}, \forall e \in \mathcal{P}_{c1-1} : e(I(h)) \subseteq I(\Theta(h, e)). \quad (16)$$

Intuitively, if a learner h learns languages L , then $\Theta(h, e)$ learns $e(L)$.

Remark 3.2. Let I be computably \mathcal{P}_{c1-1} -robust. Then we have

$$\forall e \in \mathcal{P}_{c1-1}, \forall \mathcal{L} \subseteq \mathcal{E} : \mathcal{L} \in I \Leftrightarrow e(\mathcal{L}) \in I.$$

Remark 3.3. Let $(\alpha, \beta) \in \mathbf{The30}$. Then (α, β) is computably \mathcal{P}_{c1-1} -robust.

Definition 3.4. Let $I = (\alpha, \beta)$ be a learning criterion. We call I *data normal* iff, for all p_0 such that $W_{p_0} = \emptyset$, there is a computable operator $\hat{\Theta} : \mathfrak{P} \rightarrow \mathfrak{P}$ such that

$$I(h) \subseteq I(\hat{\Theta}(h)) \quad (17)$$

and (a) – (d) below.

(a) There is $f_\beta \in \mathcal{R}$ such that

$$\forall h, T, \forall i > 0 : \beta(h, T)(i) = h(f_\beta(T[i], \beta(h, T)[i])).^{11} \quad (18)$$

(b) There is a function $d_\beta \in \mathcal{R}$ such that

$$\begin{aligned} \forall T \in \mathbf{Txt}, i \in \mathbb{N} : \beta(\hat{\Theta}(h), T)[i] \downarrow \Rightarrow \\ d_\beta(f_\beta(T[i], \beta(\hat{\Theta}(h), T)[i])) \in \begin{cases} \{\#\} & \text{if content}(T[i]) = \emptyset; \\ \text{content}(T[i]), & \text{otherwise.}^{12} \end{cases} \end{aligned} \quad (19)$$

(c) For all $h \in \mathcal{P}$,

$$\forall \sigma, \tau : d_\beta(f_\beta(\sigma, \tau)) = \# \Rightarrow \hat{\Theta}(h)(f_\beta(\sigma, \tau)) = p_0.^{13} \quad (20)$$

(d) For all $h, h' \in \mathcal{P}$,

$$[\forall L \in I(h) \forall T \in \mathbf{Txt}(L) : \beta(h, T) = \beta(h', T)] \Rightarrow I(h) \subseteq I(h').^{14} \quad (21)$$

Remark 3.5. Let $(\alpha, \beta) \in \mathbf{The30}$. Then (α, β) is data normal.

Next (Theorem 3.6) is the main result of this section, giving *sufficient* conditions for when a separation will necessarily be witnessed by a *specific* self-learning class of languages. As a corollary, we get that *for each pair of learning criteria from The30*, any separations are witnessed by such classes! In this sense, self-learning classes of languages capture the *essence* of separations (when they exist). Note that the proof of the theorem would simplify a lot, were one to suppose somewhat stronger properties of the learning criteria, in particular, excluding the use of **It** and **ItCtr** as sequence generating operators. Theorem 3.6 can be modified to cover other kinds of criteria, for example, those pertaining to learnability by total learners. In a future paper, we will analyze self learning-classes in more depth and will provide further theorems like Theorem 3.6.

¹¹Intuitively, the i -th conjecture of h on T depends only on some information (as specified by f_β) about the first i datapoints and conjectures.

¹²Intuitively, from the information given by f_β , a datum (if any) that this datum is based on can be extracted.

¹³Intuitively, constantly outputting one and the same index for the empty language is a viable strategy as long as no numerical data has been presented.

¹⁴Intuitively, changing a learner on inputs that do not present data from a language learned does not harm learnability.

Theorem 3.6. Learning criteria are as in Definition 2.1. Let I_0 and I_1 be computably \mathcal{P}_{c1-1} -robust learning criteria. Suppose I_1 is data normal as witnessed by f_1 and d_1 . Let p_0 be such that $W_{p_0} = \emptyset$ and h_1 be such that

$$\forall x : h_1(x) = \begin{cases} p_0, & \text{if } d_1(x) = \#; \\ \varphi_{d_1(x)}(x), & \text{otherwise.} \end{cases} \quad (22)$$

Further, let $\mathcal{L}_1 = I_1(h_1)$. Then we have

$$I_1 \setminus I_0 \neq \emptyset \Leftrightarrow \mathcal{L}_1 \notin I_0.$$

Proof. The implication “ \Leftarrow ” is obvious. Regarding “ \Rightarrow ”, let $\mathcal{L} \in I_1$ as witnessed by h and suppose $\mathcal{L} \notin I_0$. Let Θ be as given by I_1 being computably \mathcal{P}_{c1-1} -robust. Let $\hat{\Theta}$ be as given by I_1 being data normal. By padded **ORT**, there is a strictly monotone increasing $e \in \mathcal{R}$ such that

$$\forall x, y : \varphi_{e(x)}(y) = (\hat{\Theta} \circ \Theta)(h, e)(y). \quad (23)$$

As $e \in \mathcal{P}_{c1-1}$ and I_0 is computably \mathcal{P}_{c1-1} -robust, we have, from Remark 3.2 with I_0 in the place of I , $e(\mathcal{L}) \notin I_0$. It now suffices to show $e(\mathcal{L}) \subseteq \mathcal{L}_1$, as this would imply $\mathcal{L}_1 \notin I_0$ as desired.

Suppose $I_1 = (\alpha_1, \beta_1)$. Let $L \in e(\mathcal{L})$ and $T \in \mathbf{Txt}(L)$. We show, by induction on i ,

$$\forall i : \beta_1(h_1, T)(i) = \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)(i). \quad (24)$$

Let $h' = \Theta(h, e)$. For all i with $\text{content}(T[i]) = \emptyset$, we have

$$\begin{aligned} \beta_1(h_1, T)[i] &\stackrel{(18)}{=} h_1(f_1(T[i], \beta_1(h_1, T)[i])) \stackrel{(22)}{=} p_0 \\ &\stackrel{(19) \ \& \ (20)}{=} \hat{\Theta}(h')(f_1(T[i], \beta_1(\hat{\Theta}(h'), T)[i])) \stackrel{(18)}{=} \beta_1(\hat{\Theta}(h'), T)(i). \end{aligned} \quad (25)$$

Let $i \in \mathbb{N}$, suppose $\text{content}(T[i]) \neq \emptyset$ and (inductively) $\beta_1(h_1, T)[i] = \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)[i]$. Let

$$x = f_1(T[i], \beta_1(h_1, T)[i]) \stackrel{\text{IH}}{=} f_1(T[i], \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)[i]). \quad (26)$$

Note that $d_1(x) \stackrel{(19)}{\in} \text{content}(T[i]) \subseteq L \subseteq \text{range}(e)$. We have

$$\beta_1(h_1, T)(i) \stackrel{(18) \ \& \ (26)}{=} h_1(x) \stackrel{(22)}{=} \varphi_{d_1(x)}(x) \stackrel{(23)}{=} (\hat{\Theta} \circ \Theta)(h, e)(x) \stackrel{(18) \ \& \ (26)}{=} \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)(i). \quad (27)$$

This concludes the induction. Thus, h_1 on any text for a language from $e(\mathcal{L})$ makes the same conjectures as $(\hat{\Theta} \circ \Theta)(h, e)$ on T . By (16) and (17), $(\hat{\Theta} \circ \Theta)(h, e)$ I_1 -learns $e(\mathcal{L})$; thus, using (d) of I_1 being data normal, $e(\mathcal{L}) \subseteq I_1(h_1) = \mathcal{L}_1$. \square

4 Helping Remove U-Shapes

In this section we provide, in Theorem 4.8, a general technique for helping with the *removal* of U-shapes from a learner, *preserving what is learned*. When applicable, this shows U-shapes *unnecessary*. Theorem 4.8 is technically a characterization theorem, and is applied in Section 5 to provide cases where *strong* non U-shaped learning makes no difference. Also, in this section is another result, Theorem 4.9, which could similarly be used to provide other cases where mere non U-shaped learning makes no difference — although we do not apply this theorem in the present paper.

Lemma 4.1. Let $h \in \mathcal{P}$. Then there is a infinite set $L \in \mathcal{E}$ such that h does not **TxtGEx**-learn any $L' \supseteq L$ such that $L' =^* L$.

*Proof.*¹⁵ Trivial if $\mathbb{N} \notin \mathbf{TxtGEx}(h)$. Otherwise, let σ be a locking sequence for h on \mathbb{N} , $D = \text{content}(\sigma)$. Then, obviously, h does not learn any L such that $D \subseteq L \subset \mathbb{N}$. \square

Definition 4.2. For each $h \in \mathcal{P}$, let L_h denote a set L corresponding to h and as shown existent in Lemma 4.1.

Definition 4.3. Let $h \in \mathcal{P}$, $\mathcal{L} = \mathbf{TxtGEx}(h)$ and let Q be a ce set. Then, using padded s-m-n, there is a strictly monotone increasing function $p_{h,Q} \in \mathcal{R}$

$$\forall e, x : W_{p_{h,Q}(e,x)} = \{y \in L_h \mid Q(e, x)\} \cup \{y \in W_e \mid \text{not } (Q(e, x) \text{ in } \leq y \text{ steps})\}. \quad (28)$$

¹⁵This version of the proof is due to Frank Stephan [Ste09].

Lemma 4.4. Let $h \in \mathcal{P}$, $\mathcal{L} = \mathbf{TxtGEx}(h)$ and Q be a **ce** set. For all e, x , we have

$$W_{p_{h,Q}(e,x)} \in \mathcal{L} \Rightarrow \neg Q(e, x) \quad (29)$$

$$\Rightarrow W_{p_{h,Q}(e,x)} = W_e. \quad (30)$$

Proof. Immediate from the choice of L_h . \square

Lemma 4.5. Let $\beta \in \{\mathbf{It}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{Psd}, \mathbf{G}\}$. We have that

$$\{\langle e_0, e_1, e_2 \rangle \mid e_0 \text{ is not a } \beta\text{-sink of } \varphi_{e_1} \text{ on } W_{e_2}\} \text{ is } \mathbf{ce} \quad (31)$$

and, for all $f \in \mathcal{F}$ (where \mathcal{F} is from (10)),

$$\{\langle e_0, e_1, e_2 \rangle \mid e_0 \text{ is not an } f\text{-weak } \beta\text{-sink of } \varphi_{e_1} \text{ on } W_{e_2}\} \text{ is } \mathbf{ce}. \quad (32)$$

Proof. Immediate. \square

Remember that $\mathcal{R}_{c1-1} \subseteq \mathcal{R}$ denotes the set of all 1-1 *total* computable functions with computable domain and range.

Definition 4.6. A sequence generating operator β is called *1-1 left-modifiable* iff

$$\forall r \in \mathcal{R}_{c1-1} \exists s_l, s_r \in \mathcal{R} \forall h \in \mathcal{P}, T \in \mathbf{Txt} : \beta(s_l \circ h \circ s_r, T) = r \circ \beta(h, T).$$

Remark 4.7. $\mathbf{It}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{Psd}$ and \mathbf{G} are 1-1 left-modifiable.

We now present the two main Theorems of this section, the first of which characterizes strong non-U-shaped learning; the second is a sufficient condition on (not necessarily strong) non-U-shaped learning.

Theorem 4.8. Let β be 1-1 left-modifiable and fulfill (a) of data normal. Let $\alpha \in \{\mathbf{T}, \mathcal{R}\}$. Then

$$\mathbf{Sink}\alpha\mathbf{Txt}\beta\mathbf{Ex} = \mathbf{SNU}\alpha\mathbf{Txt}\beta\mathbf{Ex}.$$

Proof. “ \supseteq ”: Let $h_0 \in \mathcal{P}$, $\mathcal{L} = \mathbf{SNU}\mathbf{Txt}\beta\mathbf{Ex}(h_0)$. Let $L \in \mathcal{L}$ and let T be a text for L . Let k be such that h_0 has converged on T after $T[k]$ to some e . Then

$$W_e = L. \quad (33)$$

To show that e is a β -sink of h_0 on L : Let $T \in \mathbf{Txt}(L)$ and i_0 such that $\beta(h_0, T)(i_0) = e$. Let $i \geq i_0$. Then, as h_0 is strongly non-U-shaped on L and from (33), $\beta(h_0, T)(i) = e$.

“ \subseteq ”: Let $h_0 \in \mathcal{P}$, $\mathcal{L} = \mathbf{Sink}\mathbf{Txt}\beta\mathbf{Ex}(h_0)$. Let Q be a **ce** predicate such that

$$\forall e : Q(e) \Leftrightarrow e \text{ is not a } \beta\text{-sink of } h_0 \text{ on } W_e. \quad (34)$$

With f_β as given by (a) of data normal, let $h_0^* = h_0 \circ f_\beta$. Let $p = p_{h_0^*, Q}$ as in Definition 4.3. Let β 's left-modifiability with respect to p be witnessed by s_l and $s_r \in \mathcal{R}$. Let $h \in \mathcal{R}$ be such that

$$h = s_l \circ h_0 \circ s_r. \quad (35)$$

Note that, if $h_0 \in \mathcal{R}$, then $h \in \mathcal{R}$.

Claim 1. h is strongly non-U-shaped.

Proof of Claim 1. Let $L \in \mathcal{L}$, $e \in \mathbb{N}$ and σ such that $\text{content}(\sigma) \subseteq L$. Suppose $h^*(\sigma) = p(e)$ and

$$W_{p(e)} = L \in \mathcal{L}. \quad (36)$$

From (29) we get $\neg Q(e)$. Hence, from the definition of Q in (34), for all texts T for L extending σ , we have that h_0 has syntactically converged after seeing σ . Thus, h has syntactically converged after seeing σ (and, thus, does not exhibit a U-shape). \square (FOR CLAIM 1)

Claim 2. $\mathcal{L} \subseteq \mathbf{Txt}\beta\mathbf{Ex}(h)$.

Proof of Claim 2. Let $L \in \mathcal{L}$ and let T be a text for L . As h_0 is sink-locking, there is k minimal such that, with $e = h_0^*(T[k])$,

$$e \text{ is a sink of } h \text{ on } L. \quad (37)$$

Thus, h_0 converges on T to e ; therefore,

$$W_e = L. \quad (38)$$

Furthermore, h on T converges to $p(e)$ and, by (37), $\neg Q(e)$; hence,

$$W_{p(e)} \stackrel{(30)}{=} W_e \stackrel{(38)}{=} L. \quad (39)$$

□ (FOR CLAIM 2)

□

Theorem 4.9. Let \mathcal{F} be as in (10). Let β be 1-1 left-modifiable and fulfill (a) of data normal. Let $\alpha \in \{\mathbf{T}, \mathcal{R}\}$. Then

$$\bigcup_{f \in \mathcal{F}} \mathbf{Weaksink}^f \alpha \mathbf{Txt} \beta \mathbf{Ex} \subseteq \mathbf{NU} \alpha \mathbf{Txt} \beta \mathbf{Ex}.$$

The proof is analogous to that of the proof of “ \subseteq ” of Theorem 4.8.

Theorem 4.9 gives a good way of showing the non-U-shaped learnability of a class of languages: To define a $\bigcup_{f \in \mathcal{F}} \mathbf{Weaksink}^f \alpha \mathbf{Txt} \beta \mathbf{Ex}$ -learner to learn the class, one can use a strictly monotone increasing computable function $p \in \mathcal{R}$ (called a *padding function*) on two arguments such that $\forall e, x : W_{p(e,x)} = W_e$ and let $f \in \mathcal{F}$ be such that $f(p(e, x), p(e', x')) = 1$ iff $e = e'$.

5 Applications of the Techniques to The30

In this section we essentially apply general techniques presented in the two just previous sections to prove a number of results pertaining to the necessity of U-shapes in learning.

Note that [CM07] implies that $\mathcal{R}\mathbf{TxtSdEx} \subset \mathbf{TxtSdEx}$. This separation can also be shown using Theorem 3.6.

Theorem 5.1. We have

- (i) $\mathbf{TxtSdEx} = \mathbf{SNUTxtSdEx}$ and
- (ii) $\mathcal{R}\mathbf{TxtSdEx} = \mathbf{SNUR}\mathbf{TxtSdEx}$.

Our proof of this theorem involves an application of Theorem 4.8.

Theorem 5.2.

$$\mathbf{SNUR}\mathbf{TxtPsdEx} = \mathbf{TxtPsdEx}.$$

Our proof of this theorem involves an application of Theorem 4.8.

As $\mathbf{TxtGEx} = \mathbf{TxtPsdEx}$ [SR84, Ful85, Ful90, JORS99], we immediately get the following corollary, reproving a result from [BCM⁺08] and one from [CM08a].

Corollary 5.3.

$$\mathbf{NUTxtGEx} \stackrel{[\text{BCM}^+08]}{=} \mathbf{TxtGEx} \stackrel{[\text{CM08a}]}{=} \mathbf{SNUTxtGEx}.$$

From [CM08b, Theorem 2] we have $\mathbf{TxtItEx} = \mathbf{NUTxtItEx}$. Contrasting this result, we have the following theorem.

Theorem 5.4.

$$\mathbf{SNUTxtItEx} \subset \mathbf{NUTxtItEx}.$$

Our proof makes use of padded **ORT** and, as noted above, for convenience, a self-learning class simpler to work with than that from Theorem 3.6.

6 Summary and Open Problems

Summing up, the following main results within the focus of the present paper and regarding the necessity of syntactic or semantic U-shapes are shown or already known. Some of the justificatory remarks omit citing $\mathbf{TxtGEx} = \mathbf{TxtPsdEx}$ [SR84, Ful90] (and trivial inclusions).

$\mathbf{SNUTxtGEx}$	$=$	$\mathbf{NUTxtGEx}$	$=$	\mathbf{TxtGEx} ;
	[CM08a], Cor 5.3		[BCM ⁺ 08], Cor 5.3	
$\mathbf{SNURTxtGEx}$	$=$	$\mathbf{NURTxtGEx}$	$=$	$\mathbf{TxtRGEx}$;
	Thm 5.2		[BCM ⁺ 08], Thm 5.2	
$\mathbf{SNUTxtPsdEx}$	$=$	$\mathbf{NUTxtPsdEx}$	$=$	$\mathbf{TxtPsdEx}$;
	Thm 5.2		Thm 5.2	
$\mathbf{SNURTxtPsdEx}$	$=$	$\mathbf{NURTxtPsdEx}$	$=$	$\mathbf{TxtRPsdEx}$;
	Thm 5.2		Thm 5.2	
$\mathbf{SNUTxtSdEx}$	$=$	$\mathbf{NUTxtSdEx}$	$=$	$\mathbf{TxtSdEx}$;
	Thm 5.1		Thm 5.1	
$\mathbf{SNURTxtSdEx}$	$=$	$\mathbf{NURTxtSdEx}$	$=$	$\mathbf{TxtRSdEx}$;
	Thm 5.1		Thm 5.1	
$\mathbf{SNUTxtItEx}$	\subset	$\mathbf{NUTxtItEx}$	$=$	$\mathbf{TxtItEx}$.
	Thm 5.4		[CM08b]	

Trivially, we have

$$\begin{aligned} \mathbf{SNURTxtItEx} &\subseteq \mathbf{NURTxtItEx} \subseteq \mathbf{RTxtItEx}; \\ \mathbf{SNUTxtItCtrEx} &\subseteq \mathbf{NUTxtItCtrEx} \subseteq \mathbf{TxtItCtrEx}; \\ \mathbf{SNURTxtItCtrEx} &\subseteq \mathbf{NURTxtItCtrEx} \subseteq \mathbf{RTxtItCtrEx}. \end{aligned}$$

The other directions of inclusions are *open*. We think that $\mathbf{NURTxtItEx} = \mathbf{RTxtItEx}$ can be obtained as a corollary to the proof of $\mathbf{NUTxtItEx} = \mathbf{TxtItEx}$ in [CM08b]. Furthermore, we suspect we can show $\mathbf{SNURTxtItEx} \subset \mathbf{NURTxtItEx}$ by a variant of the proof above of Theorem 5.4.

Acknowledgements

We would like to thank the anonymous referees for their comments and suggestions. Further, we want to thank Samuel E. Moelius and Thomas Zeugmann for discussions on various theorems and their proofs.

References

- [BB75] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
- [BCM⁺08] G. Baliga, J. Case, W. Merkle, F. Stephan, and W. Wiehagen. When unlearning helps. *Information and Computation*, 206:694–709, 2008.
- [Car82] S. Carey. Face perception: Anomalies of development. In S. Strauss and R. Stavy, editors, *U-Shaped Behavioral Growth*, Developmental Psychology Series. Academic Press, NY, 1982.
- [Cas74] J. Case. Periodicity in generations of automata. *Mathematical Systems Theory*, 8:15–32, 1974.
- [Cas94] J. Case. Infinitary self-reference in learning theory. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:3–16, 1994.
- [Cas99] J. Case. The power of vacillation in language learning. *SIAM Journal on Computing*, 28(6):1941–1969, 1999.
- [CCJS07] L. Carlucci, J. Case, S. Jain, and F. Stephan. Non-U-shaped vacillatory and team learning. *Journal of Computer and System Sciences*, 2007. Special issue in memory of Carl Smith.
- [CJLZ99] J. Case, S. Jain, S. Lange, and T. Zeugmann. Incremental concept learning for bounded data mining. *Information and Computation*, 152:74–110, 1999.
- [CK08] J. Case and T. Kötzing. Dynamically delayed postdictive completeness and consistency in learning. In *19th International Conference on Algorithmic Learning Theory (ALT’08)*, volume 5254 of *Lecture Notes in Artificial Intelligence*, pages 389–403. Springer, 2008.
- [CK10] J. Case and T. Kötzing. Solutions to open questions for non-U-shaped learning with memory limitations, 2010. Submitted to *ALT’10*.
- [CL82] J. Case and C. Lynes. Machine inductive inference and language identification. In M. Nielsen and E. Schmidt, editors, *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, volume 140 of *Lecture Notes in Computer Science*, pages 107–115. Springer-Verlag, Berlin, 1982.

- [CM07] J. Case and S. Moelius. Parallelism increases iterative learning power. In *ALT '07: Proceedings of the 18th international conference on Algorithmic Learning Theory*, pages 49–63, Berlin, Heidelberg, 2007. Springer-Verlag.
- [CM08a] J. Case and S. Moelius. Optimal language learning. In *ALT*, volume 5254 of *Lecture Notes in Computer Science*, pages 419–433. Springer, 2008.
- [CM08b] J. Case and S. Moelius. U-shaped, iterative, and iterative-with-counter learning. *Machine Learning*, 72(1-2):63–88, 2008.
- [CS83] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
- [FJO94] M. Fulk, S. Jain, and D. Osherson. Open problems in Systems That Learn. *Journal of Computer and System Sciences*, 49(3):589–604, December 1994.
- [Ful85] M. Fulk. *A Study of Inductive Inference Machines*. PhD thesis, SUNY at Buffalo, 1985.
- [Ful90] M. Fulk. Prudence and other conditions on formal language learning. *Information and Computation*, 85:1–11, 1990.
- [Gol67] E. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [JORS99] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory*. MIT Press, Cambridge, Mass., second edition, 1999.
- [LV08] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Verlag, third edition, 2008.
- [MPU⁺92] G. Marcus, S. Pinker, M. Ullman, M. Hollander, T.J. Rosen, and F. Xu. *Overregularization in Language Acquisition*. Monographs of the Society for Research in Child Development, vol. 57, no. 4. University of Chicago Press, 1992. Includes commentary by H. Clahsen.
- [OW82] D. Osherson and S. Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.
- [RC94] J. Royer and J. Case. *Subrecursive Programming Systems: Complexity and Succinctness*. Research monograph in *Progress in Theoretical Computer Science*. Birkhäuser Boston, 1994.
- [Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967. Reprinted by MIT Press, Cambridge, Massachusetts, 1987.
- [SR84] G. Schäfer-Richter. *Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien*. PhD thesis, RWTH Aachen, 1984.
- [SS82] S. Strauss and R. Stavy, editors. *U-Shaped Behavioral Growth*. Developmental Psychology Series. Academic Press, NY, 1982.
- [Ste09] F. Stephan, 2009. Private communication.
- [TA02] N. Taatgen and J. Anderson. Why do children learn to say broke? A model of learning the past tense without feedback. *Cognition*, 86(2):123–155, 2002.
- [WC80] K. Wexler and P. Culicover. *Formal Principles of Language Acquisition*. MIT Press, Cambridge, Mass, 1980.
- [Wie76] R. Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationsverarbeitung und Kybernetik*, 12:93–99, 1976.
- [Wie91] R. Wiehagen. A thesis in inductive inference. In P. Schmitt J. Dix, K. Jantke, editor, *Nonmonotonic and Inductive Logic, 1st International Workshop*, volume 543 of *Lecture Notes in Artificial Intelligence*, pages 184–207. Springer-Verlag, Karlsruhe, Germany 1991.

Inferring Descriptive Generalisations of Formal Languages

Dominik D. Freydenberger*
Goethe Universität,
Frankfurt am Main, Germany
freydenberger@em.uni-frankfurt.de

Daniel Reidenbach
Loughborough University,
Loughborough, United Kingdom
D.Reidenbach@lboro.ac.uk

Abstract

In the present paper, we introduce a variant of Gold-style learners that is not required to infer precise descriptions of the languages in a class, but that must find descriptive patterns, i. e., optimal generalisations within a class of pattern languages. Our first main result characterises those indexed families of recursive languages that can be inferred by such learners, and we demonstrate that this characterisation shows enlightening connections to Angluin’s corresponding result for exact inference. Using a notion of descriptiveness that is restricted to the natural subclass of terminal-free E-pattern languages, we introduce a generic inference strategy, and our second main result characterises those classes of languages that can be generalised by this strategy. This characterisation demonstrates that there are major classes of languages that can be generalised in our model, but not be inferred by a normal Gold-style learner. Our corresponding technical considerations lead to deep insights of intrinsic interest into combinatorial and algorithmic properties of pattern languages.

1 Introduction

In Gold’s intensively studied learning paradigm of language identification in the limit from positive data (Gold, 1967), it is a requirement for the computational learner to infer, for any positive presentation of any language in some class, an *exact* description of that language. While this maximum accuracy of the output of the inference procedure is clearly a natural goal, it has a number of downsides, the most obvious one being the fact that it can lead to significant limitations to the learning power of the model. From a more applied point of view, there is another important reason why one might wish to relax it and settle for receiving an approximation of the language from the learner: depending on the class of languages to be inferred, the corresponding grammars or acceptors might have undesirable properties, i. e., they might have computationally hard decision problems or be incomprehensible to a (human) user. Thus, in various settings it might be perfectly acceptable for an inference procedure to output a compact and reasonably precise approximation of the language instead of producing a precise yet arbitrarily complex grammar.

In the present paper, we introduce and study such a variant of Gold’s model, where the requirement of exact language identification is dropped and replaced with that of inference of easily interpretable approximations. More precisely, we consider a learner that, for any language it reads, must converge to a *consistent pattern*, i. e., a finite string that consists of variables and of terminal symbols and that can be turned into any word of the language by substituting arbitrary strings of terminal symbols for the variables. In addition to being seen as mere descriptions of common features of words in a given language, such a pattern α can also be interpreted as a generator of a formal language $L(\alpha)$, the so-called *pattern language* (cf. Angluin (1980a)), which is simply the maximum set of words the pattern is consistent with. Hence, referring to this terminology, we can state that our learner has to output a pattern generating a language that is a superset of the input language, which means that our approach does not yield an arbitrary approximation of a language, but rather a *generalisation*. Even though many classes of pattern languages have a number of NP-complete or undecidable basic decision problems (see, e. g., Angluin (1980a), Jiang et al. (1994) and Freydenberger and Reidenbach (2010)), patterns (or related concepts, such as regular expressions and their various extensions implemented in today’s programming languages and text editors, see Câmpeanu et al. (2003)) are widely used when commonalities of words are to be specified or interpreted by a human user, which demonstrates that they are a worthwhile concept in the context of our paper.

*Corresponding author.

When inferring consistent patterns instead of precise descriptions, it is of course vital to develop and employ a notion of high-quality patterns, so that the inference procedure does not lead to an overly imprecise result. Otherwise, the learner could always output the pattern $\alpha := x_1$ (where x_1 is a variable), which is consistent with every language, and this approach would obviously neither lead to a rich theory nor to practically relevant results. In our model, the inference procedure shall therefore be required to converge to a pattern δ that is *descriptive* of the language L (with respect to a class PAT_* of pattern languages). This means that δ must be consistent with L , $L(\delta)$ must be included in PAT_* , and there is no pattern δ' satisfying $L(\delta') \in \text{PAT}_*$ and $L \subseteq L(\delta') \subset L(\delta)$; in other words, a pattern is descriptive of a language if there is no other pattern providing a closer match for the language. Since descriptiveness captures a natural understanding of patterns providing a desirable generalisation of languages and, furthermore, descriptive patterns can be used to devise Gold-style learners precisely identifying classes of pattern languages from positive data, this concept has been thoroughly investigated (see, e. g., Angluin (1980a), Jiang et al. (1994) and Freydenberger and Reidenbach (2009)). While established definitions of descriptiveness often restrict their view to patterns covering finite languages and normally use the full class of E- or NE-pattern languages (to be formally introduced in Section 2) as the class PAT_* of admissible pattern languages, we allow a descriptive pattern to cover a finite or an infinite language, and we have a class PAT_* that can be arbitrarily chosen. Both of these extensions of the original definition are absolutely straightforward.

To summarise our model of inference, we consider a learner that reads a positive presentation of a language and, after having seen a new input word, outputs a pattern, the so-called *hypothesis*. We then say that, for a class \mathcal{L} of languages and a class PAT_* of pattern languages, the learner PAT_* -*descriptively generalises* \mathcal{L} if and only if, for every positive presentation of every language $L \in \mathcal{L}$, the sequence of hypotheses produced by the learner converges to a pattern δ that is descriptive of L with respect to the class PAT_* . A more formal definition of our model is given in Section 3.1.

The main difference between descriptive generalisation and related approaches (see, e. g., Arimura et al. (1994), Mukouchi (1994), Kobayashi and Yokomori (1995), Kobayashi and Yokomori (1997) and, indirectly, Jain and Kinber (2008)) is that we have a distinct split between a class \mathcal{L} of languages to be inferred and an arbitrary class PAT_* of pattern languages determining the set of admissible hypotheses. This leads to a compact and powerful model that yields interesting insights into the question of to which extent the generalisability of \mathcal{L} depends on properties of \mathcal{L} or of PAT_* . We discuss this topic in Section 3.2, and we demonstrate in Section 3.3 that descriptive generalisation can be interpreted as a natural instance of a very general and simple inference model which, to the best of our knowledge, has not been considered so far.

In Section 4, we investigate our model for a fixed and rich class PAT_* , namely the class of *terminal-free E-pattern languages*, i. e., the class of all pattern languages generated by patterns not containing any terminal symbols, where the empty word may be substituted for the variables in the pattern. Our studies reveal that, for this choice of PAT_* , descriptive generalisation and inductive inference from positive data are incomparable, and they show that there are major and natural classes of formal languages that can be descriptively generalised according to our model, but not precisely inferred in Gold's model. Technically, our decision to focus on terminal-free E-pattern languages leads to a number of substantial combinatorial challenges for pattern languages, and we present various respective insights and tools of intrinsic interest.

Due to space constraints, Sections 2.2 and 4 do not include any proofs of formal statements.

2 Preliminaries

This paper is largely self-contained. For language theoretic and recursion theoretic notations not explicitly defined, Rozenberg and Salomaa (1997) and Rogers (1992) can be consulted, respectively.

2.1 Definitions

Let $\mathbb{N} := \{0, 1, 2, 3, \dots\}$ and let ∞ denote infinity. The symbols \subseteq , \subset , \supseteq and \supset refer to subset, proper subset, superset and proper superset relation, respectively. The symbols \mathcal{P} and \setminus denote the power set and the set difference, respectively. For an arbitrary alphabet A , a *string* (over A) is a finite sequence of symbols from A , and λ stands for the *empty string*. The symbol A^+ denotes the set of all nonempty strings over A , and $A^* := A^+ \cup \{\lambda\}$.

For any alphabet A , a *language* L (over A) is a set of strings over A , i. e. $L \subseteq A^*$. A language L is *empty* if $L = \emptyset$; otherwise, it is *nonempty*. A *class* \mathcal{L} of languages (over A) is a set of languages over A , i. e. $\mathcal{L} \subseteq \mathcal{P}(A^*)$. Let FIN_A denote the class of all finite languages over A .

For the *concatenation* of two strings w_1, w_2 we write $w_1 \cdot w_2$ or simply $w_1 w_2$. We say that a string $v \in A^*$ is a *factor* of a string $w \in A^*$ if there are $u_1, u_2 \in A^*$ such that $w = u_1 v u_2$. The notation $|K|$ stands for the size of a set K or the length of a string K ; the term $|w|_a$ refers to the number of occurrences of the symbol a in the string w . For any $w \in \Sigma^*$ and any $n \in \mathbb{N}$, w^n denotes the *n-fold concatenation* of w , with $w^0 := \lambda$. Furthermore, we use \cdot and the regular operations $*$ and $+$ on sets and strings in the usual way.

For any alphabets A, B , a *morphism* is a function $h : A^* \rightarrow B^*$ that satisfies $h(vw) = h(v)h(w)$ for all $v, w \in A^*$. Given morphisms $g : A^* \rightarrow B^*$ and $h : B^* \rightarrow C^*$ (for alphabets A, B, C), their *composition* $h \circ g$ is defined by $h \circ g(w) := h(g(w))$ for all $w \in A^*$.

A morphism $h : A^* \rightarrow B^*$ is said to be *nonerasing* if $h(a) \neq \lambda$ for all $a \in A$. For any string $w \in C^*$, where $C \subseteq A$ and $|w|_a \geq 1$ for every $a \in C$, the morphism $h : A^* \rightarrow B^*$ is called a *renaming (of w)* if $h : C^* \rightarrow B^*$ is injective and $|h(a)| = 1$ for every $a \in C$.

Let Σ be a (finite or infinite) alphabet of so-called *terminal symbols* (or: *letters*) and X an infinite set of *variables* with $\Sigma \cap X = \emptyset$. We normally assume $\{a, b, \dots\} \subseteq \Sigma$ and $\{x_1, x_2, x_3, \dots\} \subseteq X$. A *pattern* is a string over $\Sigma \cup X$, a *terminal-free pattern* is a string over X and a *word* is a string over Σ . The set of all patterns over $\Sigma \cup X$ is denoted by Pat_Σ . For any pattern α , we refer to the set of variables in α as $\text{var}(\alpha)$, and to the set of terminal symbols in α as $\text{symb}(\alpha)$.

A morphism $\sigma : (\Sigma \cup X)^* \rightarrow (\Sigma \cup X)^*$ is called *terminal-preserving* if $\sigma(a) = a$ for every $a \in \Sigma$. A terminal-preserving morphism $\sigma : (\Sigma \cup X)^* \rightarrow \Sigma^*$ is called a *substitution*.

The *NE-pattern language* $L_{\text{NE}, \Sigma}(\alpha)$ of a pattern $\alpha \in \text{Pat}_\Sigma$ is given by

$$L_{\text{NE}, \Sigma}(\alpha) := \{\sigma(\alpha) \mid \sigma : (\Sigma \cup X)^* \rightarrow \Sigma^* \text{ is a nonerasing substitution}\},$$

and the *E-pattern language* $L_{\text{E}, \Sigma}(\alpha)$ of α is given by

$$L_{\text{E}, \Sigma}(\alpha) := \{\sigma(\alpha) \mid \sigma : (\Sigma \cup X)^* \rightarrow \Sigma^* \text{ is a substitution}\}.$$

Let ePAT_Σ denote the class of all E-pattern languages over Σ , and $\text{ePAT}_{\text{tf}, \Sigma}$ the class of all terminal-free E-pattern languages over Σ . Let $\text{PAT}_{*, \Sigma}$ be a class of NE-pattern languages or a class of E-pattern languages over Σ , and let $\text{Pat}_{*, \Sigma}$ be the corresponding class of generating patterns. If the correspondence is clear, we write $L(\alpha)$ instead of $L_{\text{E}, \Sigma}(\alpha)$ or $L_{\text{NE}, \Sigma}(\alpha)$ for any $\alpha \in \text{Pat}_{*, \Sigma}$.

Let $\text{PAT}_{*, \Sigma}$ be a class of NE-pattern languages or a class of E-pattern languages over Σ . We say that a pattern $\delta \in (\Sigma \cup X)^+$ is *$\text{PAT}_{*, \Sigma}$ -descriptive* of a language $L \subseteq \Sigma^*$ if and only if $L(\delta) \in \text{PAT}_{*, \Sigma}$, $L(\delta) \supseteq L$, and there is no pattern α with $L(\alpha) \in \text{PAT}_{*, \Sigma}$ satisfying $L \subseteq L(\alpha) \subset L(\delta)$. Furthermore, $D_{\text{PAT}_{*, \Sigma}}(L)$ denotes the set of all patterns that are $\text{PAT}_{*, \Sigma}$ -descriptive of L .

Let \mathcal{L} be a class of languages over some alphabet A . Then \mathcal{L} is said to be *indexable* provided that there exists an indexing $(L_i)_{i \in \mathbb{N}}$ of languages L_i such that, first, $\mathcal{L} = \{L_i \mid i \in \mathbb{N}\}$ and, second, there exists a total computable function χ which uniformly decides the membership problem for $(L_i)_{i \in \mathbb{N}}$ – i. e., for every $w \in A^*$ and for every $i \in \mathbb{N}$, $\chi(w, i) = 1$ if and only if $w \in L_i$. In this case, we call $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ an *indexed family (of recursive languages)*. Of course, in this notation for an indexed family (which conforms with the use in the literature) the equality symbol “=” does not refer to an equality in the usual sense, but is merely a symbol indicating that \mathcal{L} contains all languages in $(L_i)_{i \in \mathbb{N}}$ and vice versa.

2.2 Preliminary Results

Obviously, the definition of a descriptive pattern is based on the inclusion of pattern languages, which is an undecidable problem for both the full class of NE-pattern languages and the full class of E-pattern languages (cf. Jiang et al. (1995), Freydenberger and Reidenbach (2010)). A significant part of our subsequent technical considerations, however, is restricted to terminal-free E-pattern languages, where the inclusion problem is known to be decidable. This directly results from the following characterisation:

Theorem 1 (Jiang et al. (1994)) *Let $|\Sigma| \geq 2$. For every $\alpha, \beta \in X^+$, $L_{\text{E}, \Sigma}(\alpha) \subseteq L_{\text{E}, \Sigma}(\beta)$ holds if and only if there is a morphism $\phi : X^* \rightarrow X^*$ with $\phi(\beta) = \alpha$.*

Unfortunately, this problem is NP-complete:

Theorem 2 (Ehrenfeucht and Rozenberg (1979)) *Let Σ be an alphabet with $|\Sigma| \geq 2$. Then the inclusion problem for $\text{ePAT}_{\text{tf}, \Sigma}$ is NP-complete.*

On the other hand, in conjunction with Reidenbach and Schneider (2009), a recent result by Holub (2009) demonstrates that the equivalence problem can be decided in polynomial time:

Theorem 3 *There is a polynomial-time algorithm deciding, for any pair of terminal-free patterns α, β and for any alphabet Σ with $|\Sigma| \geq 2$, on whether $L_{\text{E}, \Sigma}(\alpha) = L_{\text{E}, \Sigma}(\beta)$.*

As shown by Freydenberger and Reidenbach (2009), not every language has an $\text{ePAT}_{\text{tf}, \Sigma}$ - or an ePAT_Σ -descriptive pattern:

Theorem 4 *There is an infinite sequence $(\beta_n)_{n \geq 0}$ over X^+ such that, for every alphabet Σ with $|\Sigma| \geq 2$, $D_{\text{ePAT}_{\text{tf}, \Sigma}}(L_\Sigma) = D_{\text{ePAT}_\Sigma}(L_\Sigma) = \emptyset$ holds for the language $L_\Sigma := \bigcup_{n \geq 0} L_{\text{E}, \Sigma}(\beta_n)$.*

Note that L_Σ is an infinite language (and, in fact, an infinite union of languages from $\text{ePAT}_{\text{tf},\Sigma}$). In contrast to this, Jiang et al. (1994) shows that every finite language has an ePAT_Σ -descriptive pattern. This is also true when considering $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive patterns:

Proposition 5 *For every Σ with $|\Sigma| \geq 2$ and every finite nonempty $S \in \text{FIN}_\Sigma$, $D_{\text{ePAT}_{\text{tf},\Sigma}}(S) \neq \emptyset$, and a $\delta \in D_{\text{ePAT}_{\text{tf},\Sigma}}(S)$ can be effectively computed.*

While Proposition 5 proves the existence of an $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive pattern for every finite nonempty set, its proof makes use of a procedure that computes a descriptive pattern in a costly manner, since it solves the NP-complete inclusion problem (cf. Theorem 2) for an exponential number of patterns. Indeed, there is probably no algorithm that solves this problem in polynomial time:

Theorem 6 *Let Σ be an alphabet with $|\Sigma| \geq 2$. If $\text{P} \neq \text{NP}$, then there is no polynomial-time algorithm computing, for any finite set S of words, a pattern that is ePAT_Σ -descriptive of S .*

Theorem 6 addresses a problem left open by Jiang et al. (1994), and it provides a result that is stronger than the corresponding statement by Angluin (1980a) on NE-descriptive patterns. Since Theorem 6 can be proved using terminal-free patterns only, we can strengthen the corresponding result as follows:

Corollary 7 *Let Σ be an alphabet with $|\Sigma| \geq 2$. If $\text{P} \neq \text{NP}$, then there is no polynomial-time algorithm computing, for any finite set S of words, a pattern that is $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive of S .*

3 Inferring Descriptive Generalisations

In the present section, we formally introduce our notion of inferring descriptive generalisations, establish some of its basic properties (mainly by characterising, for any class of pattern languages determining the set of valid hypotheses, those indexed families that can be generalised in our model) and, finally, present a much more general inference paradigm that captures the essence of our approach. If we wish to compare Gold's well-known model of language identification in the limit from positive data (cf. Gold (1967)) with our model, then we refer to the former occasionally as LIM-TEXT. We use the same notation for the *class* of all classes of languages that can be *inferred* in that model; the meaning of this term shall therefore follow from the context.

3.1 The Inference Paradigm

We formalise our explanations on the model given in Section 1 as follows: For any alphabet Σ and any nonempty language $L \subseteq \Sigma^*$, we call a total function $t : \mathbb{N} \rightarrow \Sigma^*$ a *text* of L if and only if it satisfies $\{t(i) \mid i \in \mathbb{N}\} = L$. Moreover, for every text t and every $n \in \mathbb{N}$, t^n encodes the first n values of t in a single string, i. e. $t^n := t(1) \nabla t(2) \nabla t(3) \nabla \dots \nabla t(n)$ with $\nabla \notin \Sigma$; additionally, we define $t[n] := \{t(i) \mid i \leq n\}$. Finally, $\text{text}(L)$ denotes the set of all (computable and non-computable, repetitive and non-repetitive) texts of a language L .

Let \mathcal{L} be a class of nonempty languages over an alphabet Σ , and let $\text{PAT}_{*,\Sigma}$ be a class of NE-pattern languages or a class of E-pattern languages over Σ . Then \mathcal{L} is $\text{PAT}_{*,\Sigma}$ -*descriptively generalisable* (or, if $\text{PAT}_{*,\Sigma}$ is understood, *(descriptively) generalisable* for short) if and only if there exists a computable function $S : (\Sigma \cup \{\nabla\})^* \rightarrow (\Sigma \cup X)^+$ such that, for every $L \in \mathcal{L}$ and for every $t \in \text{text}(L)$, $S(t^n)$ is defined for every $n \in \mathbb{N}$, and there is a $\delta \in (\Sigma \cup X)^+$ with $\delta \in D_{\text{PAT}_{*,\Sigma}}(L)$ and there is an $m \in \mathbb{N}$ with $S(t^n) = \delta$ for every $n \geq m$. We call S a (*generalisation*) *strategy* and, for every $n \in \mathbb{N}$, $S(t^n)$ a *hypothesis* of S . The notation $\text{DG}_{\text{PAT}_{*,\Sigma}}$ refers to the class of all classes of languages that are $\text{PAT}_{*,\Sigma}$ -descriptively generalisable.

Consequently, and as already mentioned in Section 1, we have an inference model where the class to be inferred and the *hypothesis space* (we shall use this term in a rather informal manner for both the class $\text{PAT}_{*,\Sigma}$ and any set Pat_* of patterns satisfying $\text{PAT}_{*,\Sigma} = \{L_\Sigma(\alpha) \mid \alpha \in \text{Pat}_*\}$) are entirely different objects. We feel that this feature precisely reflects our motivation as outlined in Section 1, and it establishes the difference of our approach to a number of related models.

3.2 Fundamental Insights into the Model

We now discuss some basic properties of descriptive generalisation without considering a specific class of pattern languages determining the hypothesis space. At first glance, the definitions of descriptive generalisation and of the LIM-TEXT model are closely related, and our first observation states that they are indeed equivalent if they are applied to any class of pattern languages:

Proposition 8 *Let $\text{PAT}_{*,\Sigma}$ be a class of pattern languages. Then $\text{PAT}_{*,\Sigma} \in \text{LIM-TEXT}$ if and only if $\text{PAT}_{*,\Sigma} \in \text{DG}_{\text{PAT}_{*,\Sigma}}$.*

Proof: Directly from the definitions of LIM-TEXT and $\text{DG}_{\text{PAT}_{*,\Sigma}}$. ■

While descriptive generalisation and inductive inference from positive data, thus, seem to be very similar, there are major differences between these two models. In fact, there are classes that can be descriptively generalised, although neither the class nor the hypothesis space can be exactly inferred from positive data:

Proposition 9 *There exists a class \mathcal{L} of languages and a class $\text{PAT}_{*,\Sigma}$ of pattern languages satisfying $\mathcal{L} \notin \text{LIM-TEXT}$, $\text{PAT}_{*,\Sigma} \notin \text{LIM-TEXT}$, and $\mathcal{L} \in \text{DG}_{\text{PAT}_{*,\Sigma}}$.*

Proof: The statement follows from our Corollaries 26 and 31 in Section 4 and the fact that $\text{ePAT}_{\text{tf},\Sigma} \notin \text{LIM-TEXT}$ for $|\Sigma| = 2$ (cf. Reidenbach (2006)). ■

Since the definition of descriptive generalisation allows any class of pattern languages to be chosen as a hypothesis space, we can even devise a maximally powerful (yet utterly useless) generalisation strategy:

Proposition 10 *Let Σ be an alphabet. There exists a class $\text{PAT}_{*,\Sigma}$ of pattern languages such that every class \mathcal{L} of languages over Σ satisfies $\mathcal{L} \in \text{DG}_{\text{PAT}_{*,\Sigma}}$.*

Proof: Let $\text{PAT}_{*,\Sigma} := \{L_{E,\Sigma}(x_1)\}$. Since x_1 is $\text{PAT}_{*,\Sigma}$ -descriptive of every language $L \subseteq \Sigma^*$, a strategy S that constantly outputs x_1 generalises \mathcal{L} . ■

Obviously, the substantial gap between the LIM-TEXT model and descriptive generalisation illustrated by Proposition 10 is based on a proof that uses a trivial notion of descriptiveness. In Section 4, we shall demonstrate that there are similarly deep differences between both models if a natural and nontrivial class of pattern languages, namely $\text{ePAT}_{\text{tf},\Sigma}$, is used as admissible hypotheses for the generalisation process.

The main result of the present section is the following characterisation of descriptively generalisable indexed families of recursive languages. While our model as well as our studies in Section 4 consider descriptive generalisations of arbitrary classes of languages, this restriction facilitates an interesting comparison of our result to Angluin's characterisation of those indexed families that are inferrable in the LIM-TEXT model (see Angluin (1980b)). It is also worth noting that the subsequent argument cannot be based on strong insights into the descriptiveness of patterns, since we deal with arbitrary classes of pattern languages.

Theorem 11 *Let Σ be an alphabet, let $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ be an indexed family of nonempty recursive languages over Σ , and let $\text{PAT}_{*,\Sigma}$ be a class of pattern languages. $\mathcal{L} = (L_i)_{i \in \mathbb{N}} \in \text{DG}_{\text{PAT}_{*,\Sigma}}$ if and only if there are effective procedures d and f satisfying the following conditions:*

- (i) *For every $i \in \mathbb{N}$, there exists a $\delta_{d(i)} \in D_{\text{PAT}_{*,\Sigma}}(L_i)$ such that d enumerates a sequence of patterns $d_{i,0}, d_{i,1}, d_{i,2}, \dots$ satisfying, for all but finitely many $j \in \mathbb{N}$, $d_{i,j} = \delta_{d(i)}$.*
- (ii) *For every $i \in \mathbb{N}$, f enumerates a finite set $F_i \subseteq L_i$ such that, for every $j \in \mathbb{N}$ with $F_i \subseteq L_j$, if $\delta_{d(i)} \notin D_{\text{PAT}_{*,\Sigma}}(L_j)$, then there is a $w \in L_j$ with $w \notin F_i$.*

Proof: We begin with the *if* direction. In our proof, $F_i^{(m)}$ refers to the subset of F_i that is enumerated by f in $m \in \mathbb{N}$ steps of the computation.

We define a generalisation strategy S as follows: For any text t and for any $m \in \mathbb{N}$, when given t^m as an input, S outputs the pattern $d_{i,m}$, where $i \in \mathbb{N}$ is the smallest index satisfying: **(a)** $t[m] \subseteq L_i$ and **(b)** $F_i^{(m)} \subseteq t[m]$. If no such i exists, then S outputs $d_{0,0}$.

Since $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ is an indexed family, which means that the membership problem is uniformly decidable for all i and for all $w \in \Sigma^*$, and d and f are effective, it is obvious that S is computable and defined for every input t^m .

We now demonstrate that S $\text{PAT}_{*,\Sigma}$ -descriptively generalises $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ if (i) and (ii) are satisfied. Thus, we choose an arbitrary $n \in \mathbb{N}$ and an arbitrary text t of L_n , and we show that S , when reading t , converges to a pattern that is $\text{PAT}_{*,\Sigma}$ -descriptive of L_n . Before we start our actual reasoning, we determine a value $m_0 \in \mathbb{N}$ such that a number of vital parameters for the computation of $S(t^{m_0})$ have already stabilised: Let $m_1 \in \mathbb{N}$ be sufficiently large such that, for every $k \in \mathbb{N}$ with $k \leq n$ and $L_k \not\subseteq L_n$, $t[m_1]$ contains a word w satisfying $w \notin L_k$. The value m_1 must exist since $L_k \not\subseteq L_n$ and t is a text of L_n . Let $m_2 \in \mathbb{N}$ be sufficiently large such that, for every $k \in \mathbb{N}$ with $k \leq n$, $d_{k,m} = \delta_{d(k)}$ for every $m \geq m_2$. The value m_2 must exist due to (i). Let $m_3 \in \mathbb{N}$ be sufficiently large such that, for every $k \in \mathbb{N}$ with $k \leq n$, $F_k^{(m_3)} = F_k$. The value m_3 must exist since, according to (ii), F_k is finite. Let $m_4 \in \mathbb{N}$ be sufficiently large such that $F_n \subseteq t[m_4]$. The value m_4 must exist since t is a text of L_n and, according to (ii), $F_n \subseteq L_n$. Then $m_0 := \max\{m_1, m_2, m_3, m_4, n\}$.

Referring to these definitions, our proof of the *if* direction is based on the following Claims:

Claim 1. For every $m \geq m_0$, n satisfies $t[m] \subseteq L_n$, and $F_n^{(m)} \subseteq t[m]$.

Proof (Claim 1). The first part of the statement holds since t is a text of L_n ; the second part holds because of $m \geq m_0 \geq m_4$. ■ (Claim 1)

Claim 2. For every $m \geq m_0$, $S(t^m) \in D_{\text{PAT}_{*,\Sigma}}(L_n)$.

Proof (Claim 2). Let $S(t^m) = d_{k,m}$. By definition, S outputs the pattern $d_{k,m}$ for the smallest index $k \leq m$ satisfying conditions (a) and (b), or it outputs the auxiliary hypothesis $d_{0,0}$ if such a k does not exist. Due to Claim 1, and since $m \geq m_0$, we know that there exists at least one index (namely n) satisfying (a) and (b) for t^m . Thus, $m \geq m_0 \geq n$ implies that S does not choose to output its auxiliary hypothesis. Therefore, the following statements hold true for k : **(1)** $k \leq n$ (since S outputs $d_{k,m}$ for the smallest k satisfying conditions (a) and (b) of the definition of S); **(2)** $d_{k,m} = \delta_{d(k)}$ (because of $m \geq m_0 \geq m_2$ in conjunction with statement (1)); **(3)** $t[m] \subseteq L_k$ (because of condition (a)); **(4)** $F_k \subseteq t[m] \subseteq L_n$ (because of $m \geq m_0 \geq m_3$ in conjunction with statement (1), and due to condition (b)).

Now assume to the contrary that $S(t^m) = d_{k,m} \notin D_{\text{PAT}_{*,\Sigma}}(L_n)$. Due to statement (2), this means that $\delta_{d(k)} \notin D_{\text{PAT}_{*,\Sigma}}(L_n)$. Then statement (4) and condition (ii) of the Theorem imply that there exists a word $w \in L_n \setminus L_k$ which, due to $m \geq m_0 \geq m_1$ in conjunction with statement (1), satisfies $w \in t[m]$. Hence, $t[m] \not\subseteq L_k$. This contradicts statement (3). ■ (Claim 2)

Claim 3. There is a pattern δ and an $m' \geq m_0$ such that, for every $m \geq m'$, $S(t^m) = \delta$.

Proof (Claim 3). Due to statement (1) in the proof of Claim 2 and $m \geq m_0 \geq m_2$, there is only a finite number of possible hypotheses – namely $\delta_{d(0)}, \delta_{d(1)}, \dots, \delta_{d(n)}$ – that S can output when reading t^m . Therefore, it is sufficient to show that a hypothesis, once it has been discarded, is not chosen by S anymore. More precisely, we prove that if, for an $l_0 \geq m$, $S(t^{l_0}) = \delta_{d(k)}$ and $S(t^{l_0+1}) \neq \delta_{d(k)}$, then, for every $l \geq l_0 + 1$, there exists a $k' \neq k$ with $S(t^l) = \delta_{d(k')}$.

Since $l_0 \geq m \geq m_3$, $S(t^{l_0}) = \delta_{d(k)}$ implies **(A)** $t[l_0] \subseteq L_k$ and **(B)** $F_k \subseteq t[l_0]$. By definition, $t[l_0 + 1] \supseteq t[l_0]$, and therefore **(B)** is satisfied for t^{l_0+1} , too. Thus, the only event that can trigger a change of the hypothesis when extending t^{l_0} to t^{l_0+1} is $t[l_0 + 1] \not\subseteq L_k$; this implies $M \not\subseteq L_k$ for all supersets M of $t[l_0 + 1]$. Hence, for every $l \geq l_0 + 1$, there is a $k' \neq k$ with $S(t^l) = \delta_{d(k')}$. ■ (Claim 3)

To summarise, Claim 3 shows that S converges when reading t to a pattern δ , and Claim 2 demonstrates that δ is $\text{PAT}_{*,\Sigma}$ -descriptive of L_n . This concludes the proof of the *if* direction.

We continue with the *only if* direction. Hence, let S be a computable generalisation strategy that $\text{PAT}_{*,\Sigma}$ -descriptively generalises $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$, i. e., for every i and for every text t of L_i , S converges to a pattern that is $\text{PAT}_{*,\Sigma}$ -descriptive of L_i . We show that this implies the existence of effective procedures d and f satisfying conditions (i) and (ii).

Since $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ is an indexed family, there is an effective procedure enumerating, for every $i \in \mathbb{N}$, all words $w_{i,0}, w_{i,1}, w_{i,2}, \dots$ in L_i . Furthermore, we can use this to define a second effective procedure which enumerates, for every $i \in \mathbb{N}$, all finite sequences $s_{i,0}, s_{i,1}, s_{i,2}, \dots$ of words in L_i . Note that each sequence $s_{i,j}$, $j \in \mathbb{N}$, may contain repetitions of words. Furthermore, if L_i is finite, we can nevertheless easily make sure that the output of the above procedures is infinite for every i .

We now give a procedure that defines the behaviour of d and f :

Procedure SIM_S

Let $i \in \mathbb{N}$, and let $w_{i,0}, w_{i,1}, w_{i,2}, \dots$ and $s_{i,0}, s_{i,1}, s_{i,2}, \dots$ be as given above. Go to Stage 0.

Stage 0. Define $t_0 := w_{i,0}$, $F_i := \{w_{i,0}\}$. Define $x := 0$ and $d_{i,x} := S(t_0)$. Go to Stage 1.

Stage n ($n \geq 1$). For every $j = 0, 1, 2, \dots$ proceed as follows: Consider $s_{i,j} = (\widehat{w}_{j,0}, \widehat{w}_{j,1}, \dots, \widehat{w}_{j,y})$, $y \in \mathbb{N}$, and define $t'_j := \widehat{w}_{j,0} \nabla \widehat{w}_{j,1} \nabla \dots \nabla \widehat{w}_{j,y}$. Define $x := x + 1$ and $d_{i,x} = S(t_{n-1} \nabla t'_j)$. If $S(t_{n-1} \nabla t'_j) \neq S(t_{n-1})$, then define $t_n := t_{n-1} \nabla t'_j \nabla w_{i,n}$, $F_i := F_i \cup \{\widehat{w}_{j,0}, \widehat{w}_{j,1}, \dots, \widehat{w}_{j,y}, w_{i,n}\}$, and go to Stage $n + 1$.

Since S and the procedures enumerating the $w_{i,j}$ and $s_{i,j}$, $i, j \in \mathbb{N}$, are computable, the same holds for SIM_S. Consequently, effective procedures d and f which, for all $i \in \mathbb{N}$, uniformly produce sequences $d_{i,0}, d_{i,1}, d_{i,2}, \dots$ and enumerate F_i , respectively, can be directly derived from SIM_S.

We now show that d and f satisfy conditions (i) and (ii). Our corresponding reasoning makes use of the following fact:

Claim 4. For every $i \in \mathbb{N}$ there exists an n_0 such that procedure SIM_S, when given input i , enters Stage n_0 , but it does not enter Stage $n_0 + 1$.

Proof (Claim 4). Assume to the contrary that procedure $\text{SIM_}S$ enters an infinite number of stages. This implies that S does not converge to a fixed pattern, since $\text{SIM_}S$ goes to the next stage if and only if S changes its hypothesis for the given input. However, since all considered words are contained in L_i , each transition from Stage n to Stage $n + 1$ adds the word $w_{i,n}$ to t_n , and $\{w_{i,j} \mid j \in \mathbb{N}\} = L_i$, the string $\lim_{n \rightarrow \infty} t_n$ is an encoding of a text t of L_i . Since S $\text{PAT}_{\star, \Sigma}$ -descriptively generalises $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$, this means that S , when reading t , must converge to a pattern. This is a contradiction. ■ (Claim 4)

By definition, for every $i \in \mathbb{N}$, $\text{SIM_}S$ produces an infinite sequence of patterns. It outputs a pattern $d_{i,x+1}$ that differs from $d_{i,x}$ only if it moves from one stage to another. Thus, due to Claim 4 and for the corresponding $x_0 \in \mathbb{N}$, the sequence of patterns $d_{i,x_0}, d_{i,x_0+1}, d_{i,x_0+2}, \dots$ produced in Stage n_0 satisfies, for every $j \in \mathbb{N}$, $d_{i,x_0+j} = d_{i,x_0}$. Furthermore, due to fact that the constructed input is a text of L_i , S needs to converge to a $\text{PAT}_{\star, \Sigma}$ -descriptive pattern of L_i . This implies that $d_{i,x_0} = \delta_{d(i)}$ for a $\delta_{d(i)} \in D_{\text{PAT}_{\star, \Sigma}}(L_i)$. Consequently, the sequence of patterns $d_{i,0}, d_{i,1}, d_{i,2}, \dots$ satisfies condition (i).

$\text{SIM_}S$ adds a finite number of words to F_i if and only if it moves to the next stage. Hence, Claim 4 shows that each F_i is finite. Now assume to the contrary that, for an $i \in \mathbb{N}$, F_i does not satisfy condition (ii), i. e., there exists a $j \in \mathbb{N}$ with $F_i \subseteq L_j$, $\delta_{d(i)} \notin D_{\text{PAT}_{\star, \Sigma}}(L_j)$ and $L_j \subseteq L_i$. Let $t_{\langle j \rangle}$ be an arbitrary text of L_j . Since $F_i \subseteq L_j$ and t_{n_0-1} encodes the words in F_i , for every $m \in \mathbb{N}$, $t_{n_0-1} \nabla t_{\langle j \rangle}^m$ is an encoding of initial values of a text of L_j . Thus, for $m \rightarrow \infty$, S must converge when reading $t_{n_0-1} \nabla t_{\langle j \rangle}^m$ to a pattern that is $\text{PAT}_{\star, \Sigma}$ -descriptive of L_j . According to Claim 4, when t_{n_0-1} is continued with the encoding of any finite sequence of words from L_i , $\text{SIM_}S$ does not leave Stage n_0 . Since $L_j \subseteq L_i$, this implies that $\text{SIM_}S$ does not leave Stage n_0 for t_{n_0-1} being continued with the encoding of any finite sequence of initial values of $t_{\langle j \rangle}$. Therefore S converges, when given $t_{n_0-1} \nabla t_{\langle j \rangle}^m$ for $m = 0, 1, 2, \dots$, by definition to $\delta_{d(i)}$. This contradicts $\delta_{d(i)} \notin D_{\text{PAT}_{\star, \Sigma}}(L_j)$. Consequently, F_i satisfies condition (ii), and this concludes the proof of the *only if* direction.

Hence, $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ is $\text{PAT}_{\star, \Sigma}$ -descriptively generalisable if and only if there are effective procedures d and f satisfying conditions (i) and (ii). ■

As briefly mentioned above, Theorem 11 shows natural connections to the seminal characterisation of learnable indexed families given by Angluin (1980b), and therefore it is not surprising that some elements of our proof do not need to differ from hers. Most of these similarities result from the fact that each successful inductive inference process requires the existence of so-called *locking sequences* (see Lange et al. (2008) for a detailed discussion), and this is reflected by Angluin's *telltale* T_i and our comparable concept F_i . Nevertheless, there are crucial differences between the two characterisations. First, we need to define an enumeration of an appropriate subset of our hypothesis space (this is done by the procedure d), whereas this is automatically given in Angluin's model. In this context, it is important to note that we have to attune the set F_i to the pattern $\delta_{d(i)}$, $i \in \mathbb{N}$, which leads to d and f being defined by the same procedure $\text{SIM_}S$. Second, while Angluin's T_i must, for every j with $L_j \subset L_i$, contain a word from $L_i \setminus L_j$, our equivalent F_i only needs to do so if $\delta_{d(i)}$ is not an acceptable hypothesis for L_j . This fits with the requirement of inductive inference from positive examples to distinguish between *all* languages L_i and L_j with $L_i \neq L_j$, whereas descriptive generalisation only has to distinguish between some of them, and this requisite might be *asymmetric*, i. e., a strategy S might have to discover that a text of a language L_i is not a text of a language L_j , but it might not need to figure out that a text of L_j is not a text of L_i . The explanation of why descriptive generalisation, in general, is more powerful than inductive inference from positive data directly follows from this observation; further considerations on this topic are given in Section 3.3. Thirdly, and finally, the strategy S we deploy in our proof is, in a sense, not optimal, as it might discard a correct hypothesis – i. e. pattern $\delta_{d(j)}$ that incidentally is descriptive of the language L_i the text of which is read – simply because L_i contains a word that is not contained in L_j .

Our generic strategy S of course is not very efficient; furthermore, it has the bothersome property described above. However, it is worth mentioning that S does not test whether the given words are contained in the language of the hypothesis pattern, and it does not check the inclusion of pattern languages, either. Thus, it circumvents two decision problems that, for many natural classes of pattern languages, are known to be NP-complete or even undecidable (see, e. g., Angluin (1980a) and Freydenberger and Reidenbach (2010)), although these decision problems are essential elements of the definition of descriptiveness. Instead, S infers descriptive patterns purely based on membership tests for the languages in the indexed family. Thus, if indexed families with a fast membership test are to be generalised, then our strategy raises hope that it might be possible to do this efficiently in spite of using a hypothesis space with an NP-complete membership test. On the other hand, it might be difficult to find rich classes of pattern languages where the procedure d introduced by Theorem 11 is efficient (even though it should normally be possible to devise a d that, for every $i \in \mathbb{N}$, directly outputs the pattern $\delta_{d(i)}$ instead of enumerating the sequence $d_{i,j}$). This expectation is substantiated by Theorem 4.2 in Angluin (1980a) and our Theorem 6 and Corollary 7 given in Section 2.

3.3 A More General View

While an application of Theorem 11 might require profound knowledge on the descriptiveness of patterns, a closer look confirms our above remark that the actual characterisation and its proof do not at all. More precisely, neither the Theorem nor our reasoning deal with the properties of the descriptive patterns $\delta_{d(i)}$, $i \in \mathbb{N}$, but they merely make use of a notion of the *validity* of a hypothesis for a given language, i. e., a hypothesis is acceptable for a language if it is descriptive, but we do not check for descriptiveness. This view is quite convenient to study the difference between descriptive generalisation and inductive inference from positive data. In the LIM-TEXT model when applied to indexed families, a hypothesis i – i. e., the index of the language L_i – is valid for a language L_j , $j \neq i$, if and only if the hypothesis j is valid for the language L_i (if and only if $L_i = L_j$). In our model, this symmetry does not necessarily exist, as demonstrated by the following example:

Example 12 Let $\Sigma := \{a, b\}$. Let $L_1 := \{ababab, babab\}$ and $L_2 := \{ababab, babab, abaaba\}$. We state without proof that $\delta_1 := x_1 ababx_2$ is ePAT $_{\Sigma}$ -descriptive of L_1 and $\delta_2 := x_1x_2x_1x_2x_1$ is ePAT $_{\Sigma}$ -descriptive of L_2 . While δ_2 is also ePAT $_{\Sigma}$ -descriptive of L_1 , δ_1 is not ePAT $_{\Sigma}$ -descriptive of L_2 . Hence, a strategy S that ePAT $_{\Sigma}$ -descriptively generalises a class including L_1 and L_2 can output δ_1 or δ_2 when reading a text for L_1 , but it must not output δ_1 when reading a text for L_2 .

Referring to this phenomenon and restricted to indexed families, we can now give a much more general model of inference than the one of descriptive generalisation, and we can still characterise those indexed families that can be inferred according to this model in exactly the same way as we have done in Theorem 11. Hence, let $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ be an indexed family. Furthermore, for any $i \in \mathbb{N}$, let HYP be a function that maps i to a subset of \mathbb{N} that consists of all *valid hypotheses* for L_i . Here it is important to note that the numbers in HYP(i) do normally not refer to indices of the indexed family $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$; e. g., in our model of descriptive generalisation they would stand for indices in an arbitrary enumeration of a set of patterns. We then say that $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ is *inductively inferrable with hypotheses validity relation* HYP if and only if there exists a computable function $S : (\Sigma \cup \{\nabla\})^* \rightarrow \mathbb{N}$ such that, for every $i \in \mathbb{N}$ and for every $t \in \text{text}(L_i)$, $S(t^n)$ is defined for every $n \in \mathbb{N}$ and there is a $j \in \text{HYP}(i)$ and there is an $m \in \mathbb{N}$ with $S(t^n) = j$ for every $n \geq m$.

Our notion of descriptive generalisation demonstrates that there are natural instances of the model of inductive inference with hypotheses validity relation HYP. Nevertheless, to the best of our knowledge, its properties have not been explicitly studied so far.

As announced above, we now rephrase Theorem 11 so that it characterises those indexed families that are inductively inferrable with hypotheses validity relation HYP:

Theorem 13 Let Σ be an alphabet, let $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ be an indexed family of nonempty languages over Σ , and let HYP : $\mathbb{N} \rightarrow \mathcal{P}(\mathbb{N})$ be a function. $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ is inductively inferrable with hypotheses validity relation HYP if and only if there are effective procedures h and f satisfying the following conditions:

- (i) For every $i \in \mathbb{N}$, there exists a $\eta_i \in \text{HYP}(i)$ such that h enumerates a sequence of natural numbers i_0, i_1, i_2, \dots satisfying, for all but finitely many $k \in \mathbb{N}$, $i_k = \eta_i$.
- (ii) For every $i \in \mathbb{N}$, f enumerates a finite set $F_i \subseteq L_i$ such that, for every $j \in \mathbb{N}$ with $F_i \subseteq L_j$, if $\eta_i \notin \text{HYP}(j)$, then there is a $w \in L_j$ with $w \notin L_i$.

Proof: Minor and straightforward editing of the proof of Theorem 11 – mainly substituting h for d , i_k for $d_{i,k}$, η_i for $\delta_{d(i)}$, and HYP(i) for $D_{\text{PAT}_{*,\Sigma}}(L_i)$ – turns it into a reasoning suitable for Theorem 13. ■

To conclude this section on basic properties of our model, we wish to mention that descriptive generalisation can alternatively be interpreted as inductive inference of classes of pattern languages from *partial texts*. Hence, we can understand any language L_i as a tool to define texts that do not contain all words in $L(\delta_{d(i)})$, but nevertheless can be used to infer $\delta_{d(i)}$. Within the scope of the present paper, we do not explicitly discuss such a view, but we expect that it might be a worthwhile topic for further studies. We anticipate that its analysis might involve substantial conceptual challenges that cannot be solved using established insights into related approaches (see Fulk and Jain (1996)).

4 Inferring ePAT $_{\text{tf},\Sigma}$ -Descriptive Patterns

We now study our model for a fixed hypothesis space, namely the class ePAT $_{\text{tf},\Sigma}$. The decidability of the inclusion problem for this class (see Theorem 1) allows us to develop a set of powerful tools.

This section is divided into three parts. In the first part, we consider some questions on the existence of ePAT $_{\text{tf},\Sigma}$ -descriptive patterns for various classes of languages and develop a set of tools in order to simplify proofs on the existence and nonexistence of ePAT $_{\text{tf},\Sigma}$ -descriptive patterns.

The second part deals with a generalisation strategy that is based on the procedure that is described in Proposition 5, which we deem so natural that we call it the canonical strategy Canon for ePAT $_{\text{tf},\Sigma}$ -descriptive

generalisations. Most importantly, we give a characterisation of the class $\mathcal{TS}\mathcal{L}_\Sigma$ of languages that can be descriptively generalised with Canon.

In the final part of this section, we examine the relationship of various classes of languages to $\mathcal{TS}\mathcal{L}_\Sigma$ in order to gain further insights into $DG_{\text{ePAT}_{\text{tf},\Sigma}}$ and the power of Canon.

4.1 Basic tools

Before we proceed to an examination of $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive generalisation in the next part of this section, we develop some tools and techniques that simplify the work with $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive patterns, and gather some results on the existence and nonexistence of such patterns for some classes of languages. We begin with the following result:

Lemma 14 *Let Σ be an alphabet with $|\Sigma| \geq 2$, and let $L_1, L_2 \subseteq \Sigma^*$ with $L_1 \supseteq L_2$. If there is a $\delta \in D_{\text{ePAT}_{\text{tf},\Sigma}}(L_2)$ with $L_{\text{E},\Sigma}(\delta) \supseteq L_1$, then $\delta \in D_{\text{ePAT}_{\text{tf},\Sigma}}(L_1)$.*

This observation might seem to be elementary, but together with Lemma 17, it forms the fundament of the proof of almost every result in this section. The technical base of that Lemma derives from a phenomenon that often arises when dealing with $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive patterns. We consider the following example:

Example 15 *Let $\Sigma := \{a, b\}$ and let $L_1 := \{a^2\}$, $L_2 := \{(ab^1 a ab^2 a \dots ab^n a)^2 \mid n \geq 2\}$, and $L_3 := L_{\text{E},\Sigma}(x_1^2) \setminus \{a^2, b^2\}$. It is easy to see that all three languages are included in $L_{\text{E},\Sigma}(x_1^2)$. However, in addition to this, for every $\alpha \in X^+$ with $L_{\text{E},\Sigma}(\alpha) \supseteq L_i$ (with $1 \leq i \leq 3$), $L_{\text{E},\Sigma}(\alpha) \supseteq L_{\text{E},\Sigma}(x_1^2)$ holds as well. For L_1 , this is obvious. For L_2 , assume that $L_{\text{E},\Sigma}(\alpha) \supseteq L_2$ for some $\alpha \in X^+$, let $n := |\text{var}(\alpha)|$ and $w = (ab^1 a ab^2 a \dots ab^n a)^2 \in L_2$, and choose any morphism ϕ with $\phi(\alpha) = w$. As w contains n distinct factors of the form $ab^+ a$, each occurring exactly twice, there must be an $x \in \text{var}(\alpha)$ that contains at least one complete occurrence of such a segment, which implies $|\alpha|_x \in \{1, 2\}$. In both cases, we can construct a morphism ψ with $\psi(\alpha) = x_1^2$ (by mapping x to x_1 or x_1^2 and erasing all other variables), which (according to Theorem 1) leads to $L_{\text{E},\Sigma}(\alpha) \supseteq L_{\text{E},\Sigma}(x_1^2)$. Finally, as $L_3 \supset L_2$, this also proves the claim for L_3 .*

As $L_{\text{E},\Sigma}(x_1^2)$ and all three L_i have exactly the same superpatterns, we are able to conclude that, for every $i \in \{1, \dots, 3\}$, $D_{\text{ePAT}_{\text{tf},\Sigma}}(L_{\text{E},\Sigma}(x_1^2)) = D_{\text{ePAT}_{\text{tf},\Sigma}}(L_i)$. Although the four languages might seem rather different, they have exactly the same sets of $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive patterns.

When generalising languages using $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive patterns, every language has a certain superset that is covered by every descriptive generalisation of this language, and cannot be avoided. In order to formalise this line of reasoning (and in order to use this phenomenon), we introduce the set of *superpatterns* $\text{Super}(L)$, and the *superpattern hulls* $\text{S-Hull}_\Sigma(L)$, which are defined as

$$\begin{aligned} \text{Super}(L) &:= \{\alpha \in X^+ \mid \text{for every } w \in L, \text{ there is a morphism } \phi \text{ with } \phi(\alpha) = w\}, \\ \text{S-Hull}_\Sigma(L) &:= \bigcap_{\alpha \in \text{Super}(L)} L_{\text{E},\Sigma}(\alpha) \end{aligned}$$

for all alphabets Σ, Σ' and any language $L \subseteq (\Sigma')^*$. Note that, by Theorem 1, for every pair of patterns $\alpha, \beta \in X^+$ and every Σ with $|\Sigma| \geq 2$, $L_{\text{E},\Sigma}(\alpha) \subseteq L_{\text{E},\Sigma}(\beta)$ if and only if $\beta \in \text{Super}(L_{\text{E},\Sigma}(\alpha))$ if and only if $\beta \in \text{Super}(\{\alpha\})$. This allows us to state the following corollary:

Corollary 16 *Let Σ, Σ' be alphabets with $|\Sigma|, |\Sigma'| \geq 2$. Then $D_{\text{ePAT}_{\text{tf},\Sigma}}(L) = D_{\text{ePAT}_{\text{tf},\Sigma'}}(L)$ for every $L \subseteq (\Sigma \cap \Sigma')^*$.*

Although $\text{Super}(L)$ and $\text{S-Hull}_\Sigma(L)$ might appear to be rather simple concepts, they can be used to establish most of the results in this section. Using Lemma 14, we can develop one of our main tools:

Lemma 17 *Let Σ be an alphabet with $|\Sigma| \geq 2$. For every $L \subseteq \Sigma^*$, $D_{\text{ePAT}_{\text{tf},\Sigma}}(L) = D_{\text{ePAT}_{\text{tf},\Sigma}}(\text{S-Hull}_\Sigma(L))$.*

In a sense, $\text{S-Hull}_\Sigma(L)$ captures the whole essence of L with respect to $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive patterns, as every $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive generalisation of L is unable to distinguish between these two languages. This is illustrated by the following example:

Example 18 *Let $|\Sigma| \geq 2$ and define $L := L_{\text{E},\Sigma}(x_1^2) \cup L_{\text{E},\Sigma}(x_1^3)$. Furthermore, let*

$$\begin{aligned} \delta_1 &:= x_1^2 x_2^3, & \delta_2 &:= x_1 x_2 x_1 x_2^2, & \delta_3 &:= x_1 x_2^2 x_1 x_2, & \delta_4 &:= x_1 x_2^3 x_1, & \delta_5 &:= x_1 x_2^2 x_1^2, \\ \delta_6 &:= x_1 x_2 x_1 x_2 x_1, & \delta_7 &:= x_1 x_2 x_1^2 x_2, & \delta_8 &:= x_1^2 x_2^2 x_1, & \delta_9 &:= x_1^2 x_2 x_1 x_2, & \delta_{10} &:= x_1^3 x_2^2. \end{aligned}$$

Recalling Theorem 1, it is easy to see that, for every $\alpha \in \text{Super}(L)$, there is a δ_i , $1 \leq i \leq 10$, with $L_{\text{E},\Sigma}(\alpha) \supseteq L_{\text{E},\Sigma}(\delta_i)$ (as, for every α , there must be morphisms mapping α to both x_1^2 and x_1^3). By a

convention common in the literature, all patterns are given in canonical form (cf. Reidenbach and Schneider (2009)), where variables names are introduced in increasing lexicographic order.

This example illustrates two important phenomena. First, note that $\delta_i \in D_{\text{ePAT}_{\text{tf},\Sigma}}(L)$ for $1 \leq i \leq 10$, and for every $\delta \in D_{\text{ePAT}_{\text{tf},\Sigma}}(L)$, there is a δ_i with $L_{\text{E},\Sigma}(\delta) = L_{\text{E},\Sigma}(\delta_i)$, but $L_{\text{E},\Sigma}(\delta) \neq L_{\text{E},\Sigma}(\delta_j)$ for every $j \neq i$. Thus, L has ten distinct $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive patterns.

Second, the previous observation leads to $\text{S-Hull}_{\Sigma}(L) = \bigcap_{i=1}^{10} L_{\text{E},\Sigma}(\delta_i)$. For every $n \geq 2$, there are $j, k \geq 0$ with $n = 2j + 3k$, and therefore, $\text{S-Hull}_{\Sigma}(L) \supseteq \bigcup_{n=2}^{\infty} L_{\text{E},\Sigma}(x_1^n)$. Thus, every $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive generalisation of L is unable to exclude any language $L_{\text{E},\Sigma}(x_1^n)$ with $n \geq 2$. In this sense, $\text{S-Hull}_{\Sigma}(L)$ provides information on the coarseness of all descriptive generalisations.

Observe that L in the previous example is a finite union of languages from $\text{ePAT}_{\text{tf},\Sigma}$ that has a descriptive pattern, and recall that, according to Proposition 5, every finite set of words has an $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive pattern, while (by Theorem 4), there are infinite unions of languages from $\text{ePAT}_{\text{tf},\Sigma}$ that have no descriptive pattern.

Using Lemma 17, we can extend Proposition 5 to show that not only every finite set of words, but every finite union of languages from $\text{ePAT}_{\text{tf},\Sigma}$ has an $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive pattern:

Proposition 19 *Let Σ be an alphabet with $|\Sigma| \geq 2$, let $A = \{\alpha_1, \dots, \alpha_n\} \subset X^+$ and let $L = \bigcup_{i=1}^n L_{\text{E},\Sigma}(\alpha_i)$. Then $D_{\text{ePAT}_{\text{tf},\Sigma}}(L) \neq \emptyset$.*

Basically, Example 18 and Proposition 19 are based on the fact that words in languages from $\text{ePAT}_{\text{tf},\Sigma}$ and the generating patterns of these languages can often be used interchangeably by defining a morphism that maps the words back to their generating pattern. We proceed to develop this approach into another tool that allows us to make further statements on the (non-)existence of $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive patterns. Let $\nu : \Sigma^* \rightarrow X^*$ an arbitrary renaming. We define $\text{V-Hull}_{\Sigma}(L) := \bigcup_{w \in L} L_{\text{E},\Sigma}(\nu(w))$. Like $\text{S-Hull}_{\Sigma}(L)$, $\text{V-Hull}_{\Sigma}(L)$ is equivalent to L with respect to Super and $D_{\text{ePAT}_{\text{tf},\Sigma}}$:

Lemma 20 *Let Σ be an alphabet, $|\Sigma| \geq 2$. For every L over Σ , $\text{Super}(L) = \text{Super}(\text{V-Hull}_{\Sigma}(L))$, and $D_{\text{ePAT}_{\text{tf},\Sigma}}(L) = D_{\text{ePAT}_{\text{tf},\Sigma}}(\text{V-Hull}_{\Sigma}(L))$.*

This leads us to the following insight into the existence of $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive patterns for infinite unions of languages from $\text{ePAT}_{\text{tf},\Sigma}$:

Proposition 21 *Let $|\Sigma| \geq 2$. Then there is a set of patterns $A \subset \{x_1, x_2\}^+$ such that no pattern is $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive of $\bigcup_{\alpha \in A} L_{\text{E},\Sigma}(\alpha)$.*

Thus, unlike in the case of *finite* unions of languages from $\text{ePAT}_{\text{tf},\Sigma}$ (cf. Proposition 19), even restricting the number of variables in the generating patterns does not ensure that *infinite* unions of languages from $\text{ePAT}_{\text{tf},\Sigma}$ have a descriptive pattern. The renaming ν that maps terminals to variables can also be used to obtain the following technical result:

Lemma 22 *Let Σ be an alphabet with $|\Sigma| \geq 2$. For every nonempty language $L \subseteq \Sigma^*$, $\text{S-Hull}_{\Sigma}(L)$ is infinite.*

This insight shall be used in Section 4.3. We conclude the present part of Section 4 with a short remark illustrating that there are finite classes of languages which are not contained in $\text{DG}_{\text{ePAT}_{\text{tf},\Sigma}}$:

Proposition 23 *Let Σ be an alphabet, $|\Sigma| \geq 2$. There exists a class \mathcal{L} of nonempty languages over Σ with $|\mathcal{L}| = 1$ and $\mathcal{L} \notin \text{DG}_{\text{ePAT}_{\text{tf},\Sigma}}$.*

4.2 The Canonical Strategy and Telling Sets

According to Proposition 5, every finite set has a computable $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive pattern. We consider it the canonical strategy of descriptive inference on any text t of a given language L to compute a descriptive pattern of every initial segment t^n , in the hope that the hypotheses will converge to a pattern that is descriptive of L . As evidenced by the language $L := L_{\text{E},\Sigma}(x_1^2) \cup L_{\text{E},\Sigma}(x_1^3)$ (cf. Example 18), there are languages with more than one descriptive pattern. Furthermore, this applies also to finite languages, as for the set $S := \{a^2, b^3\}$ (for arbitrary letters $a, b \in \Sigma$), $D_{\text{ePAT}_{\text{tf},\Sigma}}(S) = D_{\text{ePAT}_{\text{tf},\Sigma}}(L)$ holds. Although S already contains all the information that is needed to compute a descriptive generalisation of L , the six distinct patterns δ_1 to δ_6 from Example 18 are all valid hypotheses. In order to allow our strategy to converge to one single hypothesis, we impose a total and well-founded order $<_{\text{LLO}}$ on X^+ and let our strategy return the $<_{\text{LLO}}$ -minimal hypothesis.

Let $<_{\text{LLO}}$ denote the length-lexicographic order¹ on X^+ . Note that $<_{\text{LLO}}$ is total and does not contain infinite decreasing chains. Thus, every set has exactly one element that is minimal with respect to $<_{\text{LLO}}$.

¹I.e., $\alpha <_{\text{LLO}} \beta$ if $|\alpha| < |\beta|$, or if $|\alpha| = |\beta|$, and α precedes β in the lexicographic order.

The strategy $\text{Canon} : (\Sigma \cup \{\nabla\})^* \rightarrow (\Sigma \cup X)^+$ is defined by, for every text t ,

$$\text{Canon}(t^n) := \delta, \text{ where } \delta \in D_{\text{ePAT}_{\text{tf},\Sigma}}(t[n]) \text{ and } \delta <_{\text{LLO}} \gamma \text{ for every other } \gamma \in D_{\text{ePAT}_{\text{tf},\Sigma}}(t[n]).$$

The computability of Canon follows immediately from the proof of Proposition 5, as all that remains is to sort the finite search space by $<_{\text{LLO}}$. We say that Canon *converges* on a text $t \in \text{text}(L)$ (of some language L over some alphabet Σ) if there is a pattern $\alpha \in X^+$ with $\text{Canon}(t^n) = \alpha$ for all but finitely many values of n . If, in addition to this, $\alpha \in D_{\text{ePAT}_{\text{tf},\Sigma}}(L)$, Canon is said to *converge correctly* on t . Now, when considering the languages L and S given in the example above, for every text $t \in \text{text}(L)$, there is an $n \geq 0$ with $S \subseteq t[n]$. From this point on, $\text{Canon}(t[n])$ will return the pattern $\delta_{10} = x_1^3 x_2^2$, as δ_{10} is an element of $(D_{\text{ePAT}_{\text{tf},\Sigma}}(S) \cap D_{\text{ePAT}_{\text{tf},\Sigma}}(L))$ and the $<_{\text{LLO}}$ -minimum of the canonical forms of the δ_i . This phenomenon leads to the definition of what we call *telling sets*, which are of crucial importance for the study of descriptive generalisability with the strategy Canon :

Definition 24 *Let $L \subseteq \Sigma^*$. A finite set $S \subseteq L$ is a telling set for L if $(D_{\text{ePAT}_{\text{tf},\Sigma}}(S) \cap D_{\text{ePAT}_{\text{tf},\Sigma}}(L)) \neq \emptyset$.*

Note that telling sets have some similarity to the concept of telltales that is used in the model of learning in the limit. For a comparison of telltales and telling sets, see our comments after Corollary 32.

Using Lemma 14, we are now able to show that the existence of a telling set is characteristic for the correct convergence of Canon on any text:

Theorem 25 *Let Σ an alphabet with $|\Sigma| \geq 2$. For every language $L \subseteq \Sigma^*$, and every text $t \in \text{text}(L)$, Canon converges correctly on t if and only if L has a telling set.*

In the final part of this section, we shall demonstrate that this is a strong result, by investigating the existence and nonexistence of telling sets for various languages.

4.3 Examination of the Class $\mathcal{TS}\mathcal{L}_\Sigma$

As stated by Theorem 25, the existence of telling sets is a strong sufficient criterion for $\text{ePAT}_{\text{tf},\Sigma}$ -descriptive generalisability. Furthermore, generalisability of a class $\mathcal{L} \subseteq \mathcal{P}(\Sigma^*)$ using Canon does not depend on the properties of the whole class, but only on the existence of a telling set for every single language $L \in \mathcal{L}$. Thus, we consider the largest possible class that can be generalised by Canon and define $\mathcal{TS}\mathcal{L}_\Sigma := \{L \subseteq \Sigma^* \mid L \text{ has a telling set}\}$. Theorem 25 immediately leads to the following corollary:

Corollary 26 *For every alphabet Σ with $|\Sigma| \geq 2$, $\mathcal{TS}\mathcal{L}_\Sigma \in \text{DG}_{\text{ePAT}_{\text{tf},\Sigma}}$.*

Thus, by examining $\mathcal{TS}\mathcal{L}_\Sigma$, we gain insights into the power of Canon and of the whole model of descriptive generalisation. Before we proceed to an examination of the relation of various classes of languages to $\mathcal{TS}\mathcal{L}_\Sigma$, we show that it is not required to choose Σ as small as possible, a result that is similar to Corollary 16, which states that $D_{\text{ePAT}_{\text{tf},\Sigma}}(L)$ is largely independent of the choice of Σ . The same holds for telling sets:

Corollary 27 *Let Σ, Σ' be alphabets with $|\Sigma|, |\Sigma'| \geq 2$. Then $L \in \mathcal{TS}\mathcal{L}_\Sigma$ if and only if $L \in \mathcal{TS}\mathcal{L}_{\Sigma'}$ for every $L \subseteq (\Sigma \cap \Sigma')$.*

This also implies that, for every $\Sigma' \supseteq \Sigma$, $\mathcal{TS}\mathcal{L}_{\Sigma'} \supseteq \mathcal{TS}\mathcal{L}_\Sigma$. We begin our examination of $\mathcal{TS}\mathcal{L}_\Sigma$ by expanding finite languages without losing their telling set properties. The next result follows immediately from Lemmas 14 and 17:

Lemma 28 *Let Σ be an alphabet with $|\Sigma| \geq 2$. Every nonempty $S \in \text{FIN}_\Sigma$ is a telling set of $\text{S-Hull}_\Sigma(S)$ and of every L with $S \subseteq L \subseteq \text{S-Hull}_\Sigma(S)$.*

In addition to showing that $\text{FIN}_\Sigma \subseteq \mathcal{TS}\mathcal{L}_\Sigma$, this result allows us (in conjunction with Lemma 22) to make the following statement on the cardinality of $\mathcal{TS}\mathcal{L}_\Sigma$:

Proposition 29 *$\mathcal{TS}\mathcal{L}_\Sigma$ is uncountable for every alphabet Σ with $|\Sigma| \geq 2$.*

This is an uncommon property, as inference from positive data is normally considered for classes consisting of countably many languages from some countable domain. Nonetheless, inferrability of uncountable classes has been studied before, see Jain et al. (2009).

Next, we shall see that $\mathcal{TS}\mathcal{L}_\Sigma$ contains a rich and natural class of languages, the DTF0L languages. A DTF0L language L over Σ is defined through a finite set of axioms $w_1, \dots, w_m \in \Sigma^*$ and a finite set of morphisms $\phi_1, \dots, \phi_n : \Sigma^* \rightarrow \Sigma^*$. Then L is the smallest language that satisfies $w_i \in L$ for every $i \in \{1, \dots, m\}$, and if $w \in L$, then $\phi_i(w) \in L$ for every $i \in \{1, \dots, n\}$. We denote the class of all DTF0L languages over Σ by DTF0L_Σ . Apart from FIN_Σ , the most prominent subclass of DTF0L_Σ is the class of D0L languages, where every language is defined through a single axiom and a single morphism (i. e., $m = n = 1$). The class D0L has been widely studied, for details, see Kari et al. (1997).

Proposition 30 *Let Σ be an alphabet with $|\Sigma| \geq 2$. Then $\text{DTF0L}_\Sigma \subseteq \text{TS}\mathcal{L}_\Sigma$.*

Lemma 28 and Proposition 30 both imply that $\text{FIN}_\Sigma \subseteq \text{TS}\mathcal{L}_\Sigma$. Furthermore, Proposition 29 and Proposition 30 both demonstrate that $\text{TS}\mathcal{L}_\Sigma$ contains at least one infinite language, which leads to the following observation:

Corollary 31 *The class $\text{TS}\mathcal{L}_\Sigma$ is superfinite for every alphabet Σ with $|\Sigma| \geq 2$.*

Together with Proposition 23, this allows us to describe the relation between $\text{DG}_{\text{ePAT}_{\text{tf},\Sigma}}$ and LIM-TEXT:

Corollary 32 *Let Σ be an alphabet, $|\Sigma| \geq 2$. Then $\text{DG}_{\text{ePAT}_{\text{tf},\Sigma}}$ and LIM-TEXT are incomparable.*

We now briefly discuss the relation between telling sets and the notion of telltales. As already mentioned in Section 3.2, according to Angluin (1980b), an indexed family $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$ of non-empty recursive languages is in LIM-TEXT if and only if there exists an effective procedure which, for every $j \geq 0$, enumerates a set T_j such that T_j is finite, $T_j \subseteq L_j$, and there does not exist a $j' \geq 0$ with $T_j \supseteq L_{j'} \supset L_j$. If there exists a set T_j satisfying these conditions, it is called a *telltale* for L_j with respect to $\mathcal{L} = (L_i)_{i \in \mathbb{N}}$. Thus, the concepts of telltales and telling sets are incomparable, as the former refers to a language and the class of languages it is contained in, whereas the latter relates to a language and certain properties of the class $\text{ePAT}_{\text{tf},\Sigma}$. Nevertheless, for every language L in $\text{ePAT}_{\text{tf},\Sigma}$, a set S is a telling set for L if and only if S is a telltale for L with respect to $\text{ePAT}_{\text{tf},\Sigma}$ (for more details on the existence of telltales for languages in $\text{ePAT}_{\text{tf},\Sigma}$, see Reidenbach (2008)).

As Proposition 33 and Proposition 34 below show, Lemma 25 by Reidenbach (2008) and Lemma 7 by Reidenbach (2006) on the existence and nonexistence of telltales lead to the corresponding results for telling sets:

Proposition 33 *Let Σ, Σ' be alphabets, $\Sigma' \subseteq \Sigma$ and $|\Sigma'| \geq 3$. For every $\alpha \in X^+$, $L_{\Sigma'}(\alpha)$ has a telling set.*

On the other hand, it is impossible to encode the structure of comparatively simple patterns in their languages with only two letters, which leads to the following negative result:

Proposition 34 *Let Σ be an alphabet with $|\Sigma| \geq 2$, and let a, b be two distinct letters from Σ . Then $L_{\Sigma, \{a,b\}}(x_1^2 x_2^2 x_3^2) \notin \text{TS}\mathcal{L}_\Sigma$.*

In contrast to this, Lemma 20 can be used to show that restricting the number of variables in the patterns leads to telling sets not only for languages from $\text{ePAT}_{\text{tf},\Sigma}$, but also for their finite unions:

Proposition 35 *Let $\alpha_1, \dots, \alpha_n \in \{x_1, \dots, x_{|\Sigma|}\}^+$, and let $L := \bigcup_{i=1}^n L_{\Sigma}(\alpha_i)$. Then $L \in \text{TS}\mathcal{L}_\Sigma$.*

Proposition 35 is especially interesting when compared to Proposition 21, which tells us that infinite unions of languages from $\text{ePAT}_{\text{tf},\Sigma}$ might not only have no telling set, but not even a descriptive pattern.

Furthermore, we state that the infinite sequence $(\beta_n)_{n \geq 0}$ that is used in the definition of the languages L_Σ for the proof of Theorem 4 describes an infinite ascending chain of languages from $\text{ePAT}_{\text{tf},\Sigma}$; i. e., $L_{\Sigma}(\beta_n) \subset L_{\Sigma}(\beta_{n+1})$ for every $n \geq 0$. Although the presence of such a chain in $\text{S-Hull}_\Sigma(L)$ for a language L does not necessarily imply emptiness of $D_{\text{ePAT}_{\text{tf},\Sigma}}(L)$, it is a sufficient criterion for $L \notin \text{TS}\mathcal{L}_\Sigma$ (again, the proof relies on Lemma 17):

Lemma 36 *Let Σ be an alphabet with $|\Sigma| \geq 2$ and let $L \subseteq \Sigma^*$. If there is an infinite chain $(\beta_n)_{n \geq 0}$ over X^+ with $L_{\Sigma}(\beta_n) \subseteq \text{S-Hull}_\Sigma(L)$ for every $n \geq 0$, $L_{\Sigma}(\beta_n) \subset L_{\Sigma}(\beta_{n+1})$ for every $n \geq 0$, and $\bigcup_{n \geq 0} L_{\Sigma}(\beta_n) \supseteq L$, then L has no telling set.*

As a direct application of this result, we can prove that there are regular languages that have no telling set:

Proposition 37 *For every alphabet Σ with $|\Sigma| \geq 2$, there is a regular language $L \subseteq \Sigma^*$ with $L \notin \text{TS}\mathcal{L}_\Sigma$.*

Note that this language is also an example of a language L that has no telling set, although $\text{S-Hull}_\Sigma(L)$ has a telling set.

Acknowledgements

The authors wish to thank Steffen Lange for his helpful remarks and suggestions on the inference model and some results presented in this paper.

References

- Angluin, D. (1980a). Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21, 46–62.
- Angluin, D. (1980b). Inductive inference of formal languages from positive data. *Information and Control*, 45, 117–135.
- Arimura, H., Shinohara, T., & Otsuki, S. (1994). Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data. *Proc. STACS 1994* (pp. 649–660).
- Câmpeanu, C., Salomaa, K., & Yu, S. (2003). A formal study of practical regular expressions. *International Journal of Foundations of Computer Science*, 14, 1007–1018.
- Ehrenfeucht, A., & Rozenberg, G. (1979). Finding a homomorphism between two words is NP-complete. *Information Processing Letters*, 9, 86–88.
- Freydenberger, D., & Reidenbach, D. (2009). Existence and nonexistence of descriptive patterns. *Proc. DLT 2009* (pp. 228–239).
- Freydenberger, D., & Reidenbach, D. (2010). Bad news on decision problems for patterns. *Information and Computation*, 208, 83–96.
- Fulk, M., & Jain, S. (1996). Learning in the presence of inaccurate information. *Theoretical Computer Science*, 161, 235–261.
- Gold, E. (1967). Language identification in the limit. *Information and Control*, 10, 447–474.
- Holub, S. (2009). Polynomial-time algorithm for fixed points of nontrivial morphisms. *Discrete Mathematics*, 309, 5069–5076.
- Jain, S., & Kinber, E. (2008). Learning and extending sublanguages. *Theoretical Computer Science*, 397, 233–246.
- Jain, S., Luo, Q., Semukhin, P., & Stephan, F. (2009). Uncountable automatic classes and learning. *Proc. ALT 2009* (pp. 293–307).
- Jiang, T., Kinber, E., Salomaa, A., Salomaa, K., & Yu, S. (1994). Pattern languages with and without erasing. *International Journal of Computer Mathematics*, 50, 147–163.
- Jiang, T., Salomaa, A., Salomaa, K., & Yu, S. (1995). Decision problems for patterns. *Journal of Computer and System Sciences*, 50, 53–63.
- Kari, L., Rozenberg, G., & Salomaa, A. (1997). L systems. In G. Rozenberg and A. Salomaa (Eds.), *Handbook of Formal Languages*, vol. 1, chapter 5, 253–328. Springer.
- Kobayashi, S., & Yokomori, T. (1995). On approximately identifying concept classes in the limit. *Proc. ALT 1995* (pp. 298–312).
- Kobayashi, S., & Yokomori, T. (1997). Learning approximately regular languages with reversible languages. *Theoretical Computer Science*, 174, 251–257.
- Lange, S., Zeugmann, T., & Zilles, S. (2008). Learning indexed families of recursive languages from positive data: A survey. *Theoretical Computer Science*, 397, 194–232.
- Mukouchi, Y. (1994). Inductive inference of an approximate concept from positive data. *Proc. ALT 1994* (pp. 484–499).
- Reidenbach, D. (2006). A non-learnable class of E-pattern languages. *Theoretical Computer Science*, 350, 91–102.
- Reidenbach, D. (2008). Discontinuities in pattern inference. *Theoretical Computer Science*, 397, 166–193.
- Reidenbach, D., & Schneider, J. (2009). Morphically primitive words. *Theoretical Computer Science*, 410, 2148–2161.
- Rogers, H. (1992). *Theory of recursive functions and effective computability*. Cambridge, MA: MIT Press. 3rd print.
- Rozenberg, G., & Salomaa, A. (1997). *Handbook of Formal Languages*, vol. 1. Berlin: Springer.

Quantum Predictive Learning and Communication Complexity with Single Input

Dmitry Gavinsky

NEC Laboratories America, Inc.
4 Independence Way, Suite 200
Princeton, NJ 08540, U.S.A.

Abstract

We define a new model of quantum learning that we call *Predictive Quantum (PQ)*. This is a quantum analogue of *PAC*, where during the testing phase the student is only required to answer a polynomial number of testing queries.

We demonstrate a relational concept class that is *efficiently learnable* in *PQ*, while in *any* “reasonable” classical model exponential amount of training data would be required. This is the first unconditional separation between quantum and classical learning.

We show that our separation is the best possible in several ways; in particular, there is no analogous result for a functional class, as well as for several weaker versions of quantum learning.

In order to demonstrate tightness of our separation we consider a special case of one-way communication that we call *single-input mode*, where Bob receives no input. Somewhat surprisingly, this setting becomes nontrivial when relational communication tasks are considered. In particular, any problem with two-sided input can be transformed into a single-input relational problem of equal *classical* one-way cost. We show that the situation is different in the *quantum* case, where the same transformation can make the communication complexity exponentially larger. This happens if and only if the original problem has exponential gap between quantum and classical one-way communication costs. We believe that these auxiliary results might be of independent interest.

1 Introduction

In this paper we compare quantum and classical modes of computational learning and give the first unconditional exponential separation between the two.

Let X be a (finite) domain and Y be a set of possible labels. Let \mathcal{C} be a *concept class* consisting of functions $\ell : X \rightarrow Y$, each $\ell \in \mathcal{C}$ can be viewed as assignment of a label to every $x \in X$. The knowledge of X , Y and \mathcal{C} is shared between a *teacher* and a *learner*; the teacher also knows some *target concept* $\ell_0 \in \mathcal{C}$, unknown to the learner. The learning process consists of two stages: the *learning phase*, followed by the *testing phase*. In the learning phase, the teacher and the learner communicate in order to let the latter learn ℓ_0 . In the testing phase, the learner has to demonstrate that he has successfully learned ℓ_0 : for example, an arbitrary $x \in X$ may be given to him, and he would have to respond with $\ell_0(x)$.

A *learning model* specifies the set of rules governing the learning and the testing phases. The teacher is, in general, viewed as an adversary that obeys the model’s restrictions.

One of the most natural and widely used learning models is that of *Probably Approximately Correct (PAC)*, defined by Valiant [V84]. In the learning phase of *PAC* a sequence of training examples

$$(x_1, \ell_0(x_1)), \dots, (x_k, \ell_0(x_k))$$

is sent by the teacher to the learner. The examples are independently chosen according to some distribution D over the domain X .¹ In the testing phase the learner is given a random $x \sim D$ and has to respond with $\ell_0(x)$.

¹Several variations of *PAC* are studied in the literature, in particular there is a definition that allows “distribution-specific” learning algorithms. In this paper we will always fix D to be the uniform distribution over X , as that is sufficient for our purposes and simplifies the notation at the same time.

Two error parameters are present in the definition of *PAC*: *accuracy* $1 - \varepsilon$ and *confidence* $1 - \delta$. We say that learning was successful if in the testing phase the learner correctly labels a randomly chosen $x \sim D$ with probability at least $1 - \varepsilon$. A learning algorithm must be successful with probability at least $1 - \delta$, taken over both algorithm's randomness and the set of examples received during the learning phase.

We say that a concept class \mathcal{C} is *efficiently learnable* in *PAC* if there exists an algorithm that runs in time at most polylogarithmic in the domain size and polynomial in $1/\varepsilon$ and $1/\delta$, and learns any $\ell \in \mathcal{C}$. Note that the running time of an algorithm is, trivially, an upper bound on the number of training examples that it uses during the learning phase.

1.1 Previous work

In [BJ95] Bshouty and Jackson introduced a natural quantum analogue of *PAC*, which we denote here by *QAC*. They gave an efficient algorithm that learns DNF formulas w.r.t. the uniform distribution from *quantum* examples – this is currently not known to be possible from classical examples (even with a quantum learning algorithm).

The question of whether quantum learning models are more efficient than the classical ones has been considered by Servedio and Gortler [SG04], who showed that the models *PAC* and *QAC* are equivalent from the information-theoretic point of view. On the other hand, they showed that quantum models are computationally more powerful than their classical analogues if certain cryptographic assumptions hold.

1.2 Our results

In the definition of a new learning model *PQ* (*Predictive Quantum*) we will generalize *QAC* in several ways.

First, we allow *relational* concept classes. Namely, the elements ℓ of \mathcal{C} can be arbitrary subsets of $X \times Y$, thus allowing multiple correct labellings for every $x \in X$. During the learning phase the learner receives pairs (x_i, y_i) , such that $x_i \sim D$ and y_i is a uniformly random element of $\{y \mid (x_i, y) \in \ell_0\}$. At the testing phase any y satisfying $(x, y) \in \ell_0$ is accepted as a correct answer to the query x .

Second, we classify all learning models as follows:

- We call *standard* a learning model where in the testing phase the learner outputs a *final hypothesis*, viewed as a function $h : X \rightarrow Y$. In the testing phase it is checked whether $h(x)$ agrees well with the target concept. The final hypothesis should be efficiently evaluatable (under the same notion of efficiency that applies to the learning algorithms in the model).
- We say that a model is *quasi-predictive* if the learner has to answer queries in the testing phase. The number of testing queries that will be asked is unknown during the learning phase.
- We call a model *predictive* if the learner should answer a single query in the testing phase.²

For example, the *PAC* model, as defined above, is predictive. If we would allow an arbitrary number of testing queries, that would make it quasi-predictive. If we require that in the end of the learning phase the learner produces a hypothesis $h : X \rightarrow Y$, such that $\Pr_{x \sim D} [h(x) = \ell(x)] \geq 1 - \varepsilon$, that turns the model into standard.

As long as the learning phase remains unchanged, standard learnability of a concept implies its quasi-predictive learnability, which, in turn, implies predictive learnability. On the other hand, it is well known that in any “reasonable” *classical* learning model, a predictive learning algorithm can be turned into a standard one (this can be achieved by producing a final hypothesis consisting of a description of the answering subroutine, all the data available after the learning phase, and a random string, if randomness is used by the answering subroutine). Therefore, in the classical case the standard, the quasi-predictive, and the predictive modes of learning are essentially equivalent; in particular, the above three definitions of *PAC* give rise to the same family of efficiently learnable concept classes. We will see that the situation is different with quantum learning.

For the rest of the paper let $n \stackrel{\text{def}}{=} \lceil \log |X| \rceil$. Consider the following definition.

Definition 1 *Let D be a distribution over X . We say that a hypothesis $h : X \rightarrow Y$ approximates a concept $\ell \in \mathcal{C}$ w.r.t. D if*

²Note that a concept class that is efficiently learnable by our definition of predictive learning is also efficiently learnable in a version where *polynomial number* of testing queries are made. For notational convenience we will use the single-query definition of predictive learnability.

- $\Pr_{x \sim D} [h(x) = \ell(x)] \geq 2/3$, when $\ell : X \rightarrow Y$ is a function;
- $\Pr_{x \sim D} [(x, h(x)) \in \ell] \geq 2/3$, when $\ell \subseteq X \times Y$ is a relation.

A hypotheses class \mathcal{H} is said to approximate \mathcal{C} if for every $\ell \in \mathcal{C}$, \mathcal{H} contains some h that approximates ℓ .

Any standard algorithm that learns \mathcal{C} with $\varepsilon \leq 1/3$ must use a class of final hypotheses that approximates \mathcal{C} . An efficient algorithm can use a class of final hypotheses of size at most exponential in $\text{poly}(n)$. As outlined above, efficient learnability in any classical model implies efficient learnability in the corresponding standard model, and therefore \mathcal{C} is *efficiently learnable in some classical model only if there exists \mathcal{H} of size at most $2^{\text{poly}(n)}$ that approximates \mathcal{C} .*

We call a concept class \mathcal{C} *unspeakable* if any class \mathcal{H} that approximates it should be of size at least $2^{2^{\Omega(n)}}$. In particular, *neither a classical algorithm nor a standard quantum algorithm can efficiently learn an unspeakable concept class.*

In this paper we demonstrate an *efficient quantum predictive algorithm that learns an unspeakable relational concept class*. Therefore, *quantum predictive learnability does not imply quantum standard learnability*. On the other hand, we will show that *no quasi-predictive quantum algorithm can efficiently learn an unspeakable concept class*. We also show that *efficient quantum learning of a functional unspeakable concept class is impossible*, and therefore the combination of relational concepts and quantum predictive mode of learning is essential for learning an unspeakable class.

Following is a summary of our main results (cf. Theorem 7, Lemma 11, and Lemma 12).

Theorem 2 *There exists a relational concept class that is unspeakable but can be efficiently learned in the model of predictive quantum PAC.*

A concept class \mathcal{C} that witnesses the above theorem is given in Definition 6. Its construction has been inspired by a communication problem due to Bar-Yossef, Jayram and Kerenidis [BJK04].

Theorem 3 *Classical learning of an unspeakable concept class is not possible from less than exponential amount of information from the teacher, even by a computationally unlimited learner.*

Both standard and quasi-predictive learning of an unspeakable concept class is not possible from less than exponential amount of quantum (w.l.g.) information from the teacher, even by a computationally unlimited learner.

Predictive learning of an unspeakable functional concept class is not possible from less than exponential amount of quantum (w.l.g.) information from the teacher, even by a computationally unlimited learner.

Two parts of Theorem 3 are proved by making connection to two “impossibility of separation” results in communication complexity. One of them is due to Aaronson [A04], and the other is new and might be of independent interest.

We will consider a special case of one-way communication, which will we call *single-input mode*, where Bob receives no input. We show that, somewhat surprisingly, for any single-input communication task the quantum and the classical one-way costs are asymptotically the same (the statement is trivial for functional tasks, but the relational case is more involved). More details can be found in Section 4.2.

2 Definitions and more

For $a \in \mathbb{N}$ we denote $[a] \stackrel{\text{def}}{=} \{1, \dots, a\}$. We view the elements of \mathbb{Z}_a as integers $\{0, 1, \dots, a-1\}$, and accordingly we define their ordering $0 < 1 < \dots < a-1$. For any $i \in \mathbb{N}$ and $b \in \mathbb{Z}_a$, let $i \cdot b = ib$ be the i 'th power of b w.r.t. the group operation $+$.

We use subscripts to address individual bits of binary strings: for $x \in \{0, 1\}^n$ and $i \in [n]$, x_i stands for the i 'th bit of x .

Let D be the uniform distribution over X , recall Definition 1.

Definition 4 *Let \mathcal{C} be a concept class. We say that \mathcal{C} is unspeakable if $|\mathcal{C}'| \in 2^{2^{\Omega(n)}}$ holds for any \mathcal{C}' that approximates \mathcal{C} w.r.t. D .*

2.1 Quantum learning

In [SG04] the authors provide an excellent survey of quantum vs. classical learning. Below we sketch one possible intuition behind the concepts considered in this work.³

Starting from *PAC*, how can we make it quantum? First, we can give the student ability to run any computation that a quantum computer can perform efficiently (e.g., to decide membership in any language from the complexity class *BQP*). Second, we can let the training examples be quantum, i.e., the student receives from the teacher *quantum bits (qubits)*. In this paper we consider the situation when *both the student and the examples are quantum*.

Information-theoretic consequences of “quantumness” stem from the facts that, on the one hand, quantum states require *exponential* (in the number of qubits) amount of classical bits for their full description, while on the other hand, the uncertainty principle dictates that given a quantum state only a (tiny) fraction of that classical data can be accessed by an observer.⁴

Note also that computational impact of a student being quantum is not necessarily captured by the power of *BQP*: As training examples are quantum, the student can apply quantum algorithms to quantum input, while *BQP* only deals with situations when quantum algorithms are fed with classical input.

What can be viewed as a reasonable model of quantum training examples? Let the target concept be ℓ_0 . First, assume that $\ell_0 : X \rightarrow Y$ is a Boolean function, then a quantum example shall look like

$$\frac{1}{\sqrt{|X|}} \sum_{x \in X} |x, \ell_0(x)\rangle,$$

where $|\cdot\rangle$ denotes the corresponding basis state⁵ of the quantum register over $n + 1$ qubits. Note that the above form of training examples corresponds to the uniform distribution of $x \in X$, since measuring the first n qubits in the computational basis can return each possible $x_0 \in X$ with the same probability of $1/|X|$.

Now, let $\ell_0 \subseteq X \times Y$ be a relation. We need a quantum superposition over all possible pairs $(x, y) \in \ell_0$. Naturally, we want to choose the amplitudes such that every x_0 still shows up with probability $1/|X|$, and at the same time, conditional on obtaining x'_0 , every element of $\{y | (x'_0, y) \in \ell_0\}$ appears with equal probability. It can be seen that the following quantum superposition satisfies the requirements:

$$\sum_{(x,y) \in \ell_0} \frac{1}{\sqrt{|X| \cdot |\{y' | (x, y') \in \ell_0\}|}} |x, y\rangle.$$

This quantum state will be used in Definition 5 below to describe the training examples that our student will receive from the teacher.

2.2 The model of predictive quantum learning

We will usually ignore normalization factors and global phases of quantum states.⁶ We define a predictive quantum version of *PAC*, as follows.

Definition 5 *In the PQ (Predictive Quantum) learning model, a learning algorithm can ask for arbitrarily many copies of the state*

$$\sum_{(x,y) \in \ell_0} \frac{1}{\sqrt{|\{y' | (x, y') \in \ell_0\}|}} |x, y\rangle,$$

³This part is mostly meant to assist a reader whose familiarity with quantum computing is limited; analyzing the philosophical foundations of quantum mechanics is beyond our scope.

⁴To visualize the uncertainty principle, consider a classical observer who wants to measure a quantum particle moving with velocity $v(t)$ and taking position $x(t)$, as functions of time t . At the moment t_0 it is possible to measure $v(t_0)$ with high accuracy, but then $x(t_0)$ can only be determined very roughly; alternatively, it is possible to measure $x(t_0)$ with high accuracy, but that leaves $v(t_0)$ with large uncertainty. The “quantum catch” here is that the uncertainty does not result from any kind of “imperfection” in the measurement devices being used, but rather constitutes one of several fundamental principles of quantum mechanics.

⁵This is Dirac’s “bra-ket” notation: *Kets* ($|\cdot\rangle$) denote unit vectors corresponding to pure quantum states, and *bras* ($\langle\cdot|$) stand for the complex conjugates of *kets*. Naturally, $\langle\cdot|\cdot\rangle$ and $|\cdot\rangle\langle\cdot|$ are, respectively, the inner and the outer products of two vector operands.

⁶That is, we allow arbitrary non-zero complex vectors to represent quantum states.

where $\ell_0 \subseteq X \times Y$ is a relational target concept. In the end of the learning process the algorithm receives an element $x \in X$ and should, with probability at least $5/6$, output any y satisfying $(x, y) \in \ell_0$.

A learning algorithm is efficient if its running time is at most polynomial in $n \stackrel{\text{def}}{=} \lceil \log |X| \rceil$. A concept class \mathcal{C} is efficiently learnable in PQ if there exists an efficient algorithm that PQ -learns every $\ell \in \mathcal{C}$.

In the above definition the relative amplitudes of the pairs $|x, y\rangle$ in a training example are chosen such that a projective measurement in the computational basis would result in a uniformly chosen x , and given x , all elements of $\{y' \mid (x, y') \in \ell_0\}$ are equally likely to come with it. Therefore, the model can be viewed as a natural quantum generalization of the relational version of PAC , as discussed in the Introduction.

The fact that all quantum training examples are the same lets us get rid of the confidence parameter (δ) in the definition of PQ (there is no such thing as “unlucky” sample of training examples). For simplicity, we choose the required accuracy (ϵ) to always be $5/6$. Note also that in the testing phase we want the learning algorithm to give a correct answer to any $x \in X$ with good probability (instead of just being able to cope with a randomly chosen x). This further simplifies the definition and also makes our result stronger (as we construct a PQ -algorithm, and do not state any lower bound against this model).

3 Concept class \mathcal{C}

We define a concept class \mathcal{C} that will be shown to be both unspeakable and efficiently PQ -learnable. Our definition has been inspired by a communication problem considered in [BJK04].

Definition 6 Let N be prime. Every concept in the class \mathcal{C} is represented by $C \in \{0, 1\}^N$. The set of queries is $[N - 1]$, represented by binary strings of length $n = \lceil \log N \rceil$. A pair $(x, b) \in \mathbb{Z}_N \times \{0, 1\}$ is a valid answer to query j w.r.t. $C \in \mathcal{C}$ if $C_x \oplus C_{x+j} = b$.

We slightly abuse the notation by viewing each $C \in \mathcal{C}$ either as a binary string of length N or as a set $\{(j, x, b) \mid (x, b) \text{ is a valid answer to } j \text{ w.r.t. } C\}$.

Theorem 7 The concept class \mathcal{C} is unspeakable. On the other hand, \mathcal{C} is efficiently learnable in PQ .

The two parts of the theorem will be proved in Sections 3.1 and 3.2, respectively. The key observation that we use to efficiently learn \mathcal{C} is the following (originating from [KW04]). Let a binary string $x \in \{0, 1\}^n$ be represented as a quantum state $|\alpha(x)\rangle = \sum (-1)^{x_i} |i\rangle$, where i ranges in $[n]$. Even though it is impossible to recover individual bits of x by measuring $|\alpha(x)\rangle$, there is something nontrivial about x that can be learned from $|\alpha(x)\rangle$. Namely, given any perfect matching M over $[n]$, it is possible to measure $|\alpha(x)\rangle$ in such a way that for some $(i, j) \in M$ the value of $x_i \oplus x_j$ would become known after the measurement. The quantum state $|\alpha(x)\rangle$ fits in $\lceil \log n \rceil$ qubits; on the other hand, it can be shown that the amount of classical information needed to allow similar type of access to x is $n^{\Omega(1)}$, and this is used to show that \mathcal{C} is unspeakable.

3.1 Efficient PQ -learning of \mathcal{C}

Our learner will need k PQ -examples in order to answer to the testing query with probability $1 - 1/2^k$, and whenever an answer is given it is correct.⁷ Fix $C \in \mathcal{C}$, then the training examples are of the form

$$|\alpha^C\rangle \stackrel{\text{def}}{=} \sum_{(j,x,i) \in C} |j, x, i\rangle.$$

The learner measures the last register of each of the k instances of $|\alpha^C\rangle$ in the basis $\{|0\rangle + |1\rangle, |0\rangle - |1\rangle\}$. With probability $1 - 1/2^k$ at least one measurement results in $|0\rangle - |1\rangle$, then the learner keeps that copy and abandons the rest (otherwise he gives up). Next, the learner measures the second register in the computational basis, thus obtaining in the first two registers

$$\sum_{(j,x_0,i) \in C} (-1)^i |j, x_0\rangle = \sum_{j \in [N-1]} (-1)^{C_{x_0} \oplus C_{x_0+j}} |j, x_0\rangle = \sum_{j \in [N-1]} (-1)^{C_{x_0+j}} |j, x_0\rangle$$

⁷If we allow a slightly modified form of training examples, where i is represented through the amplitude as $\sum_{(j,x,i) \in C} (-1)^i |j, x\rangle$, then it is possible to PQ -learn \mathcal{C} exactly from one such example.

for some $x_0 \in \mathbb{Z}_N$. Then he performs the transformation $|j, x_0\rangle \rightarrow |j + x_0, x_0\rangle$, and the state of the first register becomes

$$|\alpha_{x_0}^C\rangle \stackrel{\text{def}}{=} \sum_{j \in [N-1]} (-1)^{C_{x_0+j}} |x_0 + j\rangle = \sum_{k \in \mathbb{Z}_N \setminus \{x_0\}} (-1)^{C_k} |k\rangle.$$

At this point the learner is ready for the testing phase. Assume that a question $q \in [N-1]$ has been asked. Define the following perfect matching over $\mathbb{Z}_N \setminus \{x_0\}$:

$$m_q \stackrel{\text{def}}{=} \left\{ (x_0 + (2i+1)q, x_0 + (2i+2)q) \mid 0 \leq i \leq \frac{N-3}{2} \right\}.$$

Pairwise disjointness of the edges and the fact that x_0 is isolated follow from primality of N . The learner performs projective measurement of $|\alpha_{x_0}^C\rangle$ onto $(N-1)/2$ subspaces, each spanned by a pair of vectors $|a\rangle$ and $|b\rangle$ where a and b are connected in m_q (to make the measurement complete we add $|x_0\rangle\langle x_0|$ to it, but this outcome never occurs).

Assume that the outcome of the last measurement corresponds to the edge $(a, a+q) \in m_q$. Then the state of the register that contained $|\alpha_{x_0}^C\rangle$ becomes either $|a\rangle + |a+q\rangle$ or $|a\rangle - |a+q\rangle$, the former corresponding to $C_a \oplus C_{a+q} = 0$ and the latter to $C_a \oplus C_{a+q} = 1$. As the two states are orthogonal, the learner is able to distinguish and, respectively, answer $(a, 0)$ in the first case and $(a, 1)$ in the second, and that is a correct answer.

All quantum operations involved in the algorithm can be performed efficiently.

3.2 \mathcal{C} is unspeakable

Let us see that the concept class \mathcal{C} is unspeakable. The following proof uses some ideas from [BJK04] and [GKRW06].

Assume that \mathcal{C} is approximated by a class \mathcal{D} . Then there exists some $h_0 \in \mathcal{D}$ that simultaneously approximates at least $2^N / |\mathcal{D}|$ elements of \mathcal{C} , denote the set of those elements by \mathcal{C}_0 .

Consider the answers that h_0 gives to all possible queries $q \in [N-1]$. Denote $(x_q, i_q) \stackrel{\text{def}}{=} h_0(q)$ and let

$$Q_0 \stackrel{\text{def}}{=} \{q \mid (x_q, i_q) \text{ is a good answer to } q \text{ w.r.t. at least } 3/5 \text{ 'th of } \mathcal{C}_0 \text{ 's elements}\}.$$

Counting reveals that $|Q_0| \geq \frac{N-1}{6}$.

Let $e_q \stackrel{\text{def}}{=} (x_q, x_q + q)$ and $E_0 \stackrel{\text{def}}{=} \{e_q \mid q \in Q_0\}$. Every edge e_q corresponds to at most 2 different values of $q \in [N-1]$, therefore $|E_0| \geq \frac{N-1}{12}$. Consider a graph G_0 over N nodes, whose edges are the elements of E_0 . Observe that G_0 contains at least $\sqrt{2|E_0|} \geq \sqrt{\frac{N-1}{6}}$ non-isolated vertices.

Let $F_0 \subseteq G_0$ be a forest consisting of a spanning tree for each connected component of G_0 . Then F_0 contains at least $\sqrt{\frac{N-1}{24}}$ edges, denote them by E'_0 . Let $Q'_0 \subseteq Q_0$ be a subset of size $|E'_0|$, such that

$$E'_0 = \{e_q \mid q \in Q'_0\}.$$

View the elements of \mathcal{C} as binary strings of length N . Let us consider two probability distributions, one corresponding to uniformly choosing $C \in \mathcal{C}$ and the other corresponding to uniformly choosing $C \in \mathcal{C}_0$ – denote them by D^C and D_0^C , respectively. Then

$$\log \left(\frac{|\mathcal{C}|}{|\mathcal{C}_0|} \right) = \mathbf{H}[D^C] - \mathbf{H}[D_0^C],$$

where $\mathbf{H}[\cdot]$ denotes the binary entropy.

For every $e_q = (a, b)$ put $I_q \stackrel{\text{def}}{=} C_a \oplus C_b$, and let $J \stackrel{\text{def}}{=} (I_q)_{q \in Q'_0}$. It is straightforward from the construction of Q'_0 that if $C \sim D^C$ then the collection $\{I_q \mid q \in Q'_0\}$ consists of mutually independent unbiased Boolean random variables, and therefore $\mathbf{H}_{D^C}[J] = |Q'_0|$.

As $\mathbf{H}[C] = \mathbf{H}[J] + \mathbf{H}[C|J]$ holds w.r.t. any distribution of C ,

$$\begin{aligned}
\log\left(\frac{|C|}{|C_0|}\right) &= \mathbf{H}_{D^C}[C] - \mathbf{H}_{D_0^C}[C] = \mathbf{H}_{D^C}[J] - \mathbf{H}_{D_0^C}[J] + \mathbf{H}_{D^C}[C|J] - \mathbf{H}_{D_0^C}[C|J] \\
&\geq \mathbf{H}_{D^C}[J] - \mathbf{H}_{D_0^C}[J] = |Q'_0| - \mathbf{H}_{D_0^C}[J] \\
&\geq |Q'_0| - \sum_{q \in Q'_0} \mathbf{H}_{D_0^C}[I_q] = \sum_{q \in Q'_0} \left(1 - \mathbf{H}_{D_0^C}[I_q]\right),
\end{aligned} \tag{1}$$

where the first inequality follows from the fact that $\mathbf{H}_{D^C}[C|J] = N - |Q'_0|$, which is the maximum of $\mathbf{H}[C|J]$ under any distribution of C .

From the definition of Q_0 (and the fact that $Q'_0 \subseteq Q_0$), we know that each of $\{I_q | q \in Q'_0\}$ is at least $3/5$ -biased, therefore $\mathbf{H}_{D_0^C}[I_q] \leq \frac{49}{50}$, and (1) leads to

$$\log\left(\frac{|C|}{|C_0|}\right) \geq \frac{|Q'_0|}{50} = \frac{|E'_0|}{50} > \frac{\sqrt{N}}{250},$$

for sufficiently large N . According to our choice of h_0 ,

$$|\mathcal{D}| \geq \frac{|C|}{|C_0|} \in 2^{N^{\Omega(1)}} \subseteq 2^{2^{\Omega(n)}},$$

which means that the class \mathcal{C} is unspeakable.

4 Optimality of our separation

The model of PQ where we demonstrated learnability of \mathcal{C} is computationally feasible. But in the definition of PQ we have modified what is probably the most usual learning setting in several ways: Besides being quantum, our algorithm is *predictive*; moreover, the concept class that we learn is a *relational* one. In this section we will see that all these “enhancements” are essential in order to be able to learn an unspeakable class efficiently.

We already know that classical learning of an unspeakable class cannot be efficient. We will show that exponential amount of training data is required in order to learn a functional unspeakable concept (Lemma 11), as well as to learn any unspeakable concept in quasi-predictive setting (Lemma 12). The both results are established through making a connection to one-way communication complexity: Our proof of Lemma 11 is based on Aaronson’s [A04], and in order to prove Lemma 12 we establish a new fact about one-way communication complexity that might be of independent interest (Theorem 9, Corollary 10).

4.1 Quantum and classical one-way communication complexity

The one-way model of communication complexity is defined as follows. Let $P \subseteq X \times Y \times Z$ be a (relational) two-party communication problem. Input to P has the form $(x, y) \in X \times Y$, in the beginning it is split between two players: Alice receives x and Bob receives y . The goal is for Bob to produce $z \in Z$, such that $(x, y, z) \in P$. The players cooperate to achieve it, namely Alice sends a message m to Bob, and he outputs $z \in Z$ based on the message m and his portion of input y .

Assume for convenience that both the length of y and the length of m are functions of the lengths of x , and denote the latter by $n = \lceil \log |X| \rceil$. Both Alice and Bob are all-powerful computationally, and their goal is to solve the problem using as short m as possible. There are two versions of this model that we are interested in, namely *quantum* and *classical*. In the former the action of the players should obey the laws of quantum mechanics, in particular the message m is quantum and its “length” is measured in qubits; in the latter the message is classical and consists of bits. We let our protocols employ mixed strategies, i.e., shared randomness is allowed.

For any ε we say that a *protocol* \mathcal{T} solves P with error ε if Alice and Bob, who behave according to \mathcal{T} , produce a correct answer to every input $(x, y) \in X \times Y$ with probability at least $1 - \varepsilon$. For a distribution μ over $X \times Y$ we say that \mathcal{T} solves P with error ε w.r.t. μ if a correct answer is produced with probability at least $1 - \varepsilon$ when $(x, y) \sim \mu$. The ε -error communication cost of P is the smallest possible message length of a protocol that solves P with error ε , and ε -error communication cost w.r.t. μ is defined similarly. We say that the *bounded-error cost* of P is at most k if for any $\varepsilon \in \Omega(1)$ its ε -error cost is at most k .

Denote by $\mathcal{R}_\varepsilon^1(P)$ ($\mathcal{R}_{\mu,\varepsilon}^1(P)$) the classical one-way ε -error communication cost of P (w.r.t. μ), and by $\mathcal{R}^1(P)$ its bounded-error classical cost. Denote by $\mathcal{Q}_\varepsilon^1(P)$, $\mathcal{Q}_{\mu,\varepsilon}^1(P)$ and $\mathcal{Q}^1(P)$ the corresponding quantum analogs.

An important special case of relational communication problems are *functional* problems (partial or total). The following theorem follows readily from Theorem 6 of [A04]:

Theorem 8 [A04] *For any functional two-party communication problem $F : X \times Y \rightarrow Z$, it holds that $\mathcal{R}^1(F) \in O(\log(|Y|)\mathcal{Q}_\varepsilon^1(F) \log \mathcal{Q}_\varepsilon^1(F))$ for any $\varepsilon < 1/2 - \Omega(1)$.*

4.2 One-way communication when Bob receives no input

In this section we present a new result in communication complexity, it will be used later to prove Lemma 12.

Consider a special case of one-way communication that we call *single-input mode*, where Bob receives no input. Denote $\mathbf{0} \stackrel{\text{def}}{=} \{0\}$, and let $P \subseteq X \times \mathbf{0} \times Z$ be a communication task where Alice receives x and sends a single message m to Bob, who has to output $z \in Z$ based on the message m alone.

This setting is not as trivial as it may appear at first glance.⁸ For instance, any communication problem with two-sided input $P \subseteq X \times Y \times Z$ has a single-input analogue $P' \subseteq X \times \mathbf{0} \times Z^Y$, where Bob has to produce a list of answers to the original P w.r.t. all $y \in Y$. Namely, let

$$P'_{\mu,\varepsilon} \stackrel{\text{def}}{=} \left\{ (x, 0, (z_y)_{y \in Y}) \mid \Pr_{y \sim \mu_x} [(x, y, z_y) \in P] \geq \varepsilon \right\},$$

where μ is a distribution on $X \times Y$ and μ_x is the marginal distribution of B when $(A, B) \sim \mu$ and $A = x$. Note that for any μ and $\varepsilon \in \Omega(1)$, $\mathcal{R}^1(P'_{\mu,\varepsilon}) \leq \mathcal{R}^1(P)$, and on the other hand, by the Minimax theorem $\mathcal{R}^1(P) = \sup \{ \mathcal{R}^1(P'_{\mu,\varepsilon}) \}$, where the supremum is taken w.r.t. all possible μ and $\varepsilon \in \Omega(1)$.

In other words, $P'_{\mu,\varepsilon}$ is essentially as difficult to solve in the model of one-way *classical* communication as P is. Somewhat surprisingly, the same is not true in the case of quantum communication. More generally, below we show that for any single-input communication task the quantum and the classical one-way costs are asymptotically the same. In particular, this means that $\mathcal{Q}^1(P)$ can be exponentially smaller than $\mathcal{Q}^1(P'_{\mu,\varepsilon})$ for some $\varepsilon \in \Omega(1)$ – this happens if and only if the gap between $\mathcal{Q}^1(P)$ and $\mathcal{R}^1(P)$ is exponential (examples of such P were given in [BJK04], [GKKRW07]).

Theorem 9 *For any relational two-party communication problem $P \subseteq X \times \mathbf{0} \times Z$, any distribution μ over $x \in X$ and any $\Omega(1) < \varepsilon < 1 - \Omega(1)$, it holds that $\mathcal{R}_{\mu,\varepsilon}^1(P) \in O(\mathcal{Q}_{\mu,\varepsilon}^1(P))$.*

Corollary 10 *For any $P \subseteq X \times \mathbf{0} \times Z$, it holds that $\mathcal{R}^1(P) \in O(\mathcal{Q}^1(P))$.*

Proof: By the Minimax theorem, for every ε there exists μ such that $\mathcal{R}_\varepsilon^1(P) = \mathcal{R}_{\mu,\varepsilon}^1(P)$. ■

If P is a function then Corollary 10 is a very trivial special case of Theorem 8. On the other hand, Corollary 10 applies to the much more general case of relational problems, where a statement analogous to Theorem 8 provably does not hold.

Proof:(Theorem 9) Let \mathcal{W} be a valid $\mathcal{Q}_{\mu,\varepsilon}^1$ -protocol of cost m for P , i.e., \mathcal{W} guarantees error at most ε w.r.t. $x \sim \mu$. We want to build an $\mathcal{R}_{\mu,\varepsilon}^1$ -protocol of cost $O(m)$.

Let A and B be random variables taking the value of Alice's input $x \in X$ and Bob's answer $z \in Z$, respectively. Assume $A \sim \mu$ and let μ^B be the corresponding distribution of B . Conditional upon $A = x$ let $B \sim \mu_x^B$. Define a random variable B' as a "refined version" of B , namely: if $A = x$ then the conditional distribution of B' is

$$\mu_x^{B'}(z) \stackrel{\text{def}}{=} \begin{cases} \mu_x^B(z)/(1 - \varepsilon_x) & \text{if } (x, 0, z) \in P \\ 0 & \text{otherwise} \end{cases},$$

where $1 - \varepsilon_x$ is the probability that \mathcal{W} returns a correct answer on input x .

By the Holevo bound and the information processing principle,

$$m \geq \mathbf{I}[A : B] = \mathbf{E}_{A=x} [d_{KL}(\mu_x^B \parallel \mu^B)].$$

⁸It is important that we consider relational problems, for functions the single-input mode is indeed uninteresting.

For every $x \in X$,

$$\begin{aligned} d_{KL} \left(\mu_x^{B'} \middle| \middle| \mu^B \right) &= \sum_z \mu_x^{B'}(z) \log \frac{\mu_x^{B'}(z)}{\mu^B(z)} \\ &\leq \frac{1}{1 - \varepsilon_x} \sum_z \mu_x^B(z) \log \left(\frac{\mu_x^B(z)}{\mu^B(z)} \cdot \frac{1}{1 - \varepsilon_x} \right) \\ &\leq \frac{1}{1 - \varepsilon} \cdot d_{KL} \left(\mu_x^B \middle| \middle| \mu^B \right) + \frac{1}{1 - \varepsilon} \log \frac{1}{1 - \varepsilon}. \end{aligned}$$

By linearity of expectation,

$$\mathbf{E}_{A=x} \left[d_{KL} \left(\mu_x^{B'} \middle| \middle| \mu^B \right) \right] \leq \frac{m}{1 - \varepsilon} + \frac{1}{1 - \varepsilon} \log \frac{1}{1 - \varepsilon} < \frac{2m}{1 - \varepsilon}, \quad (2)$$

for sufficiently large m .

We claim that there exists an $\mathcal{R}_{\mu, \varepsilon}^1$ -protocol for P of cost $\left\lceil \frac{11m}{\varepsilon(1-\varepsilon)} \right\rceil$. By the definition of B' , any z in the support of $\mu_x^{B'}$ is a correct answer to $x \in X$. The key observation is that $\mu_x^{B'}$ is not too far from μ^B , by (2). Therefore, if Alice and Bob sample sufficiently many elements from μ^B , with high probability at least one of them would belong to the support of $\mu_x^{B'}$. Such sampling can be performed by the players locally, using shared randomness. Then Alice can send a pointer to an element which is a good answer w.r.t. her x .

Let us estimate the probability that a randomly chosen $z \sim \mu^B$ satisfies $\mu_x^{B'}(z) > 0$. Let

$$X' \stackrel{\text{def}}{=} \left\{ x \in X \middle| d_{KL} \left(\mu_x^{B'} \middle| \middle| \mu^B \right) < \frac{5m}{\varepsilon(1-\varepsilon)} \right\},$$

then it follows from (2) that $\mu(X') > 1 - \varepsilon/2$. Fix any $x_0 \in X'$ and let $Z' \stackrel{\text{def}}{=} \left\{ z \middle| \mu_{x_0}^{B'}(z) > 0 \right\}$. From

$$\sum_{z \in Z'} \mu_{x_0}^{B'}(z) \log \frac{\mu_{x_0}^{B'}(z)}{\mu^B(z)} = d_{KL} \left(\mu_{x_0}^{B'} \middle| \middle| \mu^B \right) < \frac{5m}{\varepsilon(1-\varepsilon)}$$

it follows that

$$\Pr_{z \sim \mu_{x_0}^{B'}} \left[\frac{\mu_{x_0}^{B'}(z)}{\mu^B(z)} < 2^{\frac{10m}{\varepsilon(1-\varepsilon)}} \right] \geq \frac{1}{2}.$$

Let $Z'' \stackrel{\text{def}}{=} \left\{ z \in Z' \middle| \mu^B(z) > \mu_{x_0}^{B'}(z) \cdot 2^{-\frac{10m}{\varepsilon(1-\varepsilon)}} \right\}$, then $\mu^B(Z'') > 2^{-\frac{10m}{\varepsilon(1-\varepsilon)} - 1}$.

We have that for any $x_0 \in X'$,

$$\Pr_{z \sim \mu^B} [(x_0, 0, z) \in P] = \mu^B(Z') \geq \mu^B(Z'') > 2^{-\frac{10m}{\varepsilon(1-\varepsilon)} - 1}.$$

If we sample $M \stackrel{\text{def}}{=} \left\lceil 2^{\frac{11m}{\varepsilon(1-\varepsilon)}} \right\rceil$ elements from μ^B then with probability greater than $1 - \varepsilon/3$ at least one of them is a correct answer w.r.t. to the given x_0 , whenever $x_0 \in X'$. As the latter happens with probability at least $1 - \varepsilon/2$, the unconditional probability that one of the M elements is a correct answer is greater than $1 - \varepsilon$. A pointer to one of M elements requires $\left\lceil \frac{11m}{\varepsilon(1-\varepsilon)} \right\rceil$ bits, and that is the cost of our $\mathcal{R}_{\mu, \varepsilon}^1$ -protocol for P , as required. ■

4.3 Connection to learnability of unspeakable concepts

Let us see how Theorem 8 and Corollary 10 imply that our construction in Theorem 2 is tight. First, let us see that no unspeakable *functional* concept class can be efficiently learned even in a quantum predictive learning model.

Lemma 11 *Predictive learning of an unspeakable functional concept class is not possible from less than exponential amount of quantum (w.l.g.) information from the teacher, even by a computationally unlimited learner.*

Proof: Assume that for some functional concept class \mathcal{F} that is unspeakable, the following holds. A teacher \mathcal{T} knows some $f_0 \in \mathcal{F}$, hidden from a learner \mathcal{S} . Then \mathcal{T} exchanges at most k_q qubits with \mathcal{S} . Finally, \mathcal{S} is given some x_0 from the domain X of the functions in \mathcal{F} , and is able to compute $f_0(x_0)$ with confidence at least $5/6$.

Consider the following two-party communication task \mathcal{G} . Alice receives $f_0 \in \mathcal{F}$, Bob receives $x_0 \in X$ and they have to output $f_0(x_0)$. Clearly, $\mathcal{Q}_{5/6}^1(\mathcal{G}) \leq k_q$.

Let $k_c = \mathcal{R}^1(\mathcal{G})$. As \mathcal{F} is unspeakable, $k_c \in 2^{\Omega(n)}$. By Theorem 8, $k_c \in O(n \cdot k_q \log(k_q))$, and so $k_q \in 2^{\Omega(n)}$, as required. ■

Now we show that unspeakable concepts cannot be efficiently learned in the *quasi-predictive* (or standard) setting:

Lemma 12 *Both standard and quasi-predictive learning of an unspeakable concept class is not possible from less than exponential amount of quantum (w.l.g.) information from the teacher, even by a computationally unlimited learner.*

Proof: It is enough to prove the statement only for quasi-predictive learning, and the standard model can be viewed as a special case.

Let \mathcal{C} be an unspeakable concept class consisting of relations over $X \times Y$, assume that it is learnable in the quasi-predictive model by a protocol of cost k_q . Then there exists a protocol, according to which a teacher \mathcal{T} who knows some $\ell_0 \in \mathcal{C}$ exchanges at most k_q qubits with a learner \mathcal{S} who doesn't know ℓ_0 . Nevertheless, afterward \mathcal{S} is able to answer with sufficient confidence any number of testing questions regarding ℓ_0 .

For us it is enough to consider the testing phase where all possible $x \in X$ are asked (say, in the lexicographic order) and the learner responds with $(y_x)_{x \in X}$, such that

$$\forall (\ell_0, x) \in \mathcal{C} \times X : \Pr [(x, y_x) \in \ell_0] \geq 5/6,$$

where the probability is taken w.r.t. possible runs of the learning protocol for the given $\ell_0 \in \mathcal{C}$.

Define a relational single-input communication problem $P_{\mathcal{C}} \subseteq \mathcal{C} \times \mathbf{0} \times Y^X$ as

$$P_{\mathcal{C}} \stackrel{\text{def}}{=} \left\{ (\ell_0, 0, (y_x)_{x \in X}) \mid |\{x \mid (x, y_x) \in \ell_0\}| \geq \frac{4}{5} |X| \right\}.$$

The learning protocol for \mathcal{C} that we considered above can be turned into a \mathcal{Q}^1 -protocol of cost k_q for $P_{\mathcal{C}}$ that is correct with probability $1 - o(1)$ w.r.t. every $\ell_0 \in \mathcal{C}$, in particular $\mathcal{Q}^1(P_{\mathcal{C}}) \leq k_q$. By Corollary 10, $\mathcal{R}^1(P_{\mathcal{C}}) \in O(k_q)$.

Any \mathcal{R}^1 -protocol of cost k_c for $P_{\mathcal{C}}$ readily leads to an approximating class for \mathcal{C} of size 2^{k_c} . As \mathcal{C} is unspeakable, $k_c \in 2^{\Omega(n)}$, where $n = \log |X|$. Therefore, $k_q \in 2^{\Omega(n)}$, as required. ■

For simplicity, in the two proofs above we assumed distribution-free mode of learning, where the learner in the testing phase had to give correct answer to any $x \in X$ with high probability. Distributional versions of Lemmas 11 and 12 can be proved similarly.

5 Open problems

We demonstrated that efficient quantum predictive learning of an unspeakable relational concept class is possible. The following questions seem interesting.

When we considered the limitations of quantum quasi-predictive learning (in the proof of Lemma 12), we argued that certain “quasi-hypothesis” of polynomial length can be extracted from an efficient quantum quasi-predictive learning algorithm. But our construction does not rely upon the efficiency of the learning algorithm, and on the other hand, the quasi-hypothesis we construct cannot, in general, be efficiently evaluated. It would be interesting to come up with a stronger argument that would “preserve efficiency”; or otherwise, to give an example of an interesting quantum quasi-predictive learning algorithm. Similar observations can be made w.r.t. our proof of Lemma 11. The transformation in [A04] is, in general, not efficient. Are there interesting quantum predictive (or even quasi-predictive) learning algorithms for functional concepts?

In the above questions by “interesting” we meant quantum algorithms for learning a concept class that *admits* concise hypotheses, but only those that cannot be efficiently evaluated. Observe that such “quasi-unspeakable” concept classes cannot be learned efficiently in any reasonable classical model (in the classical setting the equivalence between standard and predictive learning is efficiency-preserving).

Note that a trivial positive answer to these questions would follow, e.g., from an assumption that $BQP \not\subseteq P/poly$. Therefore the goal should be to weaken the assumptions.

More generally, give new examples of efficient quantum (quasi-)predictive learning of concept classes that are not efficiently learnable classically. Such examples might be interesting even for models stronger than PQ (e.g., one may allow the learner to make *membership queries*).

Acknowledgments

I thank Rahul Jain for helpful discussions.

References

- [A04] S. Aaronson. Limitations of Quantum Advice and One-Way Communication. *Proceedings of the 19th IEEE Conference on Computational Complexity*, pages 320-332, 2004.
- [BJ95] N. Bshouty and J. Jackson. Learning DNF over the Uniform Distribution using a Quantum Example Oracle. *Proceedings of the 8th Annual Conference on Computational Learning Theory*, pages 118-127, 1995.
- [BJK04] Z. Bar-Yossef, T. S. Jayram and I. Kerenidis. Exponential Separation of Quantum and Classical One-Way Communication Complexity. *Proceedings of 36th Symposium on Theory of Computing*, pages 128-137, 2004.
- [GKKRW07] D. Gavinsky, J. Kempe, I. Kerenidis, R. Raz and R. de Wolf. Exponential Separations for One-Way Quantum Communication Complexity, with Applications to Cryptography. *Proceedings of the 39th Symposium on Theory of Computing*, pages 516-525, 2007.
- [GKRW06] D. Gavinsky, J. Kempe, O. Regev and R. de Wolf. Bounded-error Quantum State Identification and Exponential Separations in Communication Complexity. *Proceedings of the 38th Symposium on Theory of Computing*, pages 594-603, 2006.
- [KW04] I. Kerenidis and R. de Wolf. Exponential Lower Bound for 2-Query Locally Decodable Codes via a Quantum Argument. *Journal of Computer and System Sciences*, 69(3), pages 395-420, 2004.
- [SG04] R. Servedio and S. Gortler. Equivalences and Separations Between Quantum and Classical Learnability. *SIAM Journal on Computing* 33(5), pages 1067-1092, 2004.
- [V84] L. Valiant. A Theory of Learnable. *Communications of the ACM* 27(11), pages 1134-1142, 1984.

Online Learning of Noisy Data with Kernels

Nicolò Cesa-Bianchi
Università degli Studi di Milano
cesa-bianchi@dsi.unimi.it

Shai Shalev Shwartz
The Hebrew University
shais@cs.huji.ac.il

Ohad Shamir
The Hebrew University
ohadsh@cs.huji.ac.il

Abstract

We study online learning when individual instances are corrupted by adversarially chosen random noise. We assume the noise distribution is unknown, and may change over time with no restriction other than having zero mean and bounded variance. Our technique relies on a family of unbiased estimators for non-linear functions, which may be of independent interest. We show that a variant of online gradient descent can learn functions in any dot-product (e.g., polynomial) or Gaussian kernel space with any analytic convex loss function. Our variant uses randomized estimates that need to query a random number of noisy copies of each instance, where with high probability this number is upper bounded by a constant. Allowing such multiple queries cannot be avoided: Indeed, we show that online learning is in general impossible when only one noisy copy of each instance can be accessed.

1 Introduction

In many machine learning applications training data are typically collected by measuring certain physical quantities. Examples include bioinformatics, medical tests, robotics, and remote sensing. These measurements have errors that may be due to several reasons: sensor costs, communication constraints, or intrinsic physical limitations. In all such cases, the learner trains on a distorted version of the actual “target” data, which is where the learner’s predictive ability is eventually evaluated. In this work we investigate the extent to which a learning algorithm can achieve a good predictive performance when training data are corrupted by noise with unknown distribution.

We prove upper and lower bounds on the learner’s cumulative loss in the framework of online learning, where examples are generated by an arbitrary and possibly adversarial source. We model the measurement error via a random perturbation which affects each instance observed by the learner. We do not assume any specific property of the noise distribution other than zero-mean and bounded variance. Moreover, we allow the noise distribution to change at every step in an adversarial way and fully hidden from the learner. Our positive results are quite general: by using a randomized unbiased estimate for the loss gradient and a randomized feature mapping to estimate kernel values, we show that a variant of online gradient descent can learn functions in any dot-product (e.g., polynomial) or Gaussian RKHS under any given analytic convex loss function. Our techniques are readily extendable to other kernel types as well.

In order to obtain unbiased estimates of loss gradients and kernel values, we allow the learner to query a random number of independently perturbed copies of the current unseen instance. We show how low-variance estimates can be computed using a number of queries that is *constant* with high probability. This is in sharp contrast with standard averaging techniques which attempts to directly estimate the noisy instance, as these require a sample whose size depends on the scale of the problem. Finally, we formally show that learning is impossible, even without kernels, when only one perturbed copy of each instance can be accessed. This is true for essentially any reasonable loss function.

Our paper is organized as follows. In the next subsection we discuss related work. In Sec. 2 we introduce our setting and justify some of our choices. In Sec. 4 we present our main results but before that, in Sec. 3, we discuss the techniques used to obtain them. In the same section, we also explain why existing techniques are insufficient to deal with our problem. The detailed proofs and subroutine implementations appear in Sec. 5, with some of the more technical lemmas and proofs relegated to [7]. We wrap up with a discussion on possible avenues for future work in Sec. 6.

1.1 Related Work

In the machine learning literature, the problem of learning from noisy examples, and, in particular, from noisy training instances, has traditionally received a lot of attention —see, for example, the recent survey [12]. On the other hand, there are comparably few theoretically-principled studies on this topic. Two of them focus on models quite different from the one studied here: random attribute noise in PAC boolean learning [3, 9], and malicious noise [10, 5]. In the first case, learning is restricted to classes of boolean functions and the noise must be independent across each boolean coordinate. In the second case, an adversary is allowed to perturb a small fraction of the training examples in an arbitrary way, making learning impossible in a strong informational sense unless this perturbed fraction is very small (of the order of the desired accuracy for the predictor).

The previous work perhaps closest to the one presented here is [11], where binary classification mistake bounds are proven for the online Winnow algorithm in the presence of attribute errors. Similarly to our setting, the sequence of instances observed by the learner is chosen by an adversary. However, in [11] the noise is generated by an adversary, who may change the value of each attribute in an arbitrary way. The final mistake bound, which only applies when the noiseless data sequence is linearly separable without kernels, depends on the sum of all adversarial perturbations.

2 Setting

We consider a setting where the goal is to predict values $y \in \mathbb{R}$ based on instances $\mathbf{x} \in \mathbb{R}^d$. In this paper we focus on kernel-based linear predictors of the form $\mathbf{x} \mapsto \langle \mathbf{w}, \Psi(\mathbf{x}) \rangle$, where Ψ is a feature mapping into some reproducing kernel Hilbert space (RKHS). We assume there exists a kernel function that efficiently implements dot products in that space, i.e., $k(\mathbf{x}, \mathbf{x}') = \langle \Psi(\mathbf{x}), \Psi(\mathbf{x}') \rangle$. Note that a special case of this setting is linear kernels, where $\Psi(\cdot)$ is the identity mapping and $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$.

The standard online learning protocol for linear prediction with kernels is defined as follows: at each round t , the learner picks a linear hypothesis \mathbf{w}_t from the RKHS. The adversary then picks an example (\mathbf{x}_t, y_t) and reveals it to the learner. The loss suffered by the learner is $\ell(\langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle, y_t)$, where ℓ is a known and fixed loss function. The goal of the learner is to minimize *regret* with respect to a fixed convex set of hypotheses \mathcal{W} , namely

$$\sum_{t=1}^T \ell(\langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle, y_t) - \min_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^T \ell(\langle \mathbf{w}, \Psi(\mathbf{x}_t) \rangle, y_t).$$

Typically, we wish to find a strategy for the learner, such that no matter what is the adversary's strategy of choosing the sequence of examples, the expression above is sub-linear in T .

We now make the following twist, which limits the information available to the learner: instead of receiving (\mathbf{x}_t, y_t) , the learner observes y_t and is given access to an *oracle* A_t . On each call, A_t returns an independent copy of $\mathbf{x}_t + Z_t$, where Z_t is a zero-mean random vector with some known finite bound on its variance (in the sense that $\mathbb{E}[\|Z_t\|^2] \leq a$ for some uniform constant a). In general, the distribution of Z_t is unknown to the learner. It might be chosen by the adversary, and change from round to round or even between consecutive calls to A_t . Note that here we assume that y_t remains unperturbed, but we emphasize that this is just for simplicity - our techniques can be readily extended to deal with noisy values as well.

The learner may call A_t more than once. In fact, as we discuss later on, being able to call A_t more than once is necessary for the learner to have any hope to succeed. On the other hand, if the learner calls A_t an unlimited number of times, it can reconstruct \mathbf{x}_t arbitrarily well by averaging, and we are back to the standard learning setting. In this paper we focus on learning algorithms that call A_t only a small, essentially constant number of times, which depends only on our choice of loss function and kernel (rather than T , the norm of \mathbf{x}_t , or the variance of Z_t , which will happen with naïve averaging techniques). Moreover, since the number of queries is bounded with very high probability, one can even produce an algorithm with an absolute bound on the number of queries, which will fail or introduce some bias with an arbitrarily small probability. For simplicity, we ignore these issues in this paper.

In this setting, we wish to minimize the regret in hindsight with respect to the unperturbed data and averaged over the noise introduced by the oracle, namely

$$\mathbb{E} \left[\sum_{t=1}^T \ell(\langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle, y_t) - \min_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^T \ell(\langle \mathbf{w}, \Psi(\mathbf{x}_t) \rangle, y_t) \right] \quad (1)$$

where the random quantities are the predictors $\mathbf{w}_1, \mathbf{w}_2, \dots$ generated by the learner, which depend on the observed noisy instances (in [7], we briefly discuss alternative regret measures, and why

they are unsatisfactory). This kind of regret is relevant where we actually wish to learn from data, without the noise causing a hindrance. In particular, consider the batch setting, where the examples $\{(\mathbf{x}_t, y_t)\}_{t=1}^T$ are actually sampled i.i.d. from some unknown distribution, and we wish to find a predictor which minimizes the expected loss $\mathbb{E}[\ell(\langle \mathbf{w}, \mathbf{x} \rangle, y)]$ with respect to new examples (\mathbf{x}, y) . Using standard online-to-batch conversion techniques, if we can find an online algorithm with a sublinear bound on Eq. (1), then it is possible to construct learning algorithms for the batch setting which are robust to noise. That is, algorithms generating a predictor \mathbf{w} with close to minimal expected loss $\mathbb{E}[\ell(\langle \mathbf{w}, \mathbf{x} \rangle, y)]$ among all $\mathbf{w} \in \mathcal{W}$.

While our techniques are quite general, the exact algorithmic and theoretical results depend a lot on which loss function and kernel is used. Discussing the loss function first, we will assume that $\ell(\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle, y)$ is a convex function of \mathbf{w} for each example (\mathbf{x}, y) . Somewhat abusing notation, we assume the loss can be written either as $\ell(\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle, y) = f(y\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle)$ or as $\ell(\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle, y) = f(\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle - y)$ for some function f . We refer to the first type as *classification losses*, as it encompasses most reasonable losses for classification, where $y \in \{-1, +1\}$ and the goal is to predict the label. We refer to the second type as *regression losses*, as it encompasses most reasonable regression losses, where y takes arbitrary real values. For simplicity, we present some of our results in terms of classification losses, but they all hold for regression losses as well with slight modifications.

We present our results under the assumption that the loss function is “smooth”, in the sense that $\ell'(a)$ can be written as $\sum_{n=0}^{\infty} \gamma_n a^n$, for any a in its domain. This assumption holds for instance for the squared loss $\ell(a) = a^2$, the exponential loss $\ell(a) = \exp(a)$, and smoothed versions of loss functions such as the hinge loss and the absolute loss (we discuss examples in more details in Subsection 4.2). This assumption can be relaxed under certain conditions, and this is further discussed in Subsection 3.2.

Turning to the issue of kernels, we note that the general presentation of our approach is somewhat hampered by the fact that it needs to be tailored to the kernel we use. In this paper, we focus on two families of kernels:

Dot Product Kernels: the kernel $k(\mathbf{x}, \mathbf{x}')$ can be written as a function of $\langle \mathbf{x}, \mathbf{x}' \rangle$. Examples of such kernels $k(\mathbf{x}, \mathbf{x}')$ are linear kernels $\langle \mathbf{x}, \mathbf{x}' \rangle$; homogeneous polynomial kernels $(\langle \mathbf{x}, \mathbf{x}' \rangle)^n$, inhomogeneous polynomial kernels $(1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^n$; exponential kernels $e^{\langle \mathbf{x}, \mathbf{x}' \rangle}$; binomial kernels $(1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^{-\alpha}$, and more (see for instance [15, 17]).

Gaussian Kernels: $k(\mathbf{x}, \mathbf{x}') = e^{-\|\mathbf{x} - \mathbf{x}'\|^2 / \sigma^2}$ for some $\sigma^2 > 0$.

Again, we emphasize that our techniques are extendable to other kernel types as well.

3 Techniques

Our results are based on two key ideas: the use of online gradient descent algorithms, and construction of unbiased gradient estimators in the kernel setting. The latter is based on a general method to build unbiased estimators for non-linear functions, which may be of independent interest.

3.1 Online Gradient Descent

There exist well developed theory and algorithms for dealing with the standard online learning setting, where the example (\mathbf{x}_t, y_t) is revealed after each round, and for general convex loss functions. One of the simplest and most well known ones is the online gradient descent algorithm due to Zinkevich [18]. Since this algorithm forms a basis for our algorithm in the new setting, we briefly review it below (as adapted to our setting).

The algorithm initializes the classifier $\mathbf{w}_1 = 0$. At round t , the algorithm predicts according to \mathbf{w}_t , and updates the learning rule according to $\mathbf{w}_{t+1} = P(\mathbf{w}_t - \eta_t \nabla_t)$, where η_t is a suitably chosen constant which might depend on t ; $\nabla_t = \ell'(y_t \langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle) y_t \Psi(\mathbf{x}_t)$ is the *gradient* of $\ell(y_t \langle \mathbf{w}, \Psi(\mathbf{x}_t) \rangle)$ with respect to \mathbf{w}_t ; and P is a projection operator on the convex set \mathcal{W} , on whose elements we wish to achieve low regret. In particular, if we wish to compete with hypotheses of bounded squared norm $B_{\mathbf{w}}$, P simply involves rescaling the norm of the predictor so as to have squared norm at most $B_{\mathbf{w}}$. With this algorithm, one can prove regret bounds with respect to any $\mathbf{w} \in \mathcal{W}$.

A “folklore” result about this algorithm is that in fact, we do not need to update the predictor by the gradient at each step. Instead, it is enough to update by some random vector of bounded variance, which merely equals the gradient in expectation. This is a useful property in settings where (\mathbf{x}_t, y_t) is not revealed to the learner, and has been used before, such as in the online bandit setting (see for instance [6, 8, 1]). Here, we will use this property in a new way, in order to devise algorithms which are robust to noise. When the kernel and loss function are linear (e.g., $\Psi(\mathbf{x}) = \mathbf{x}$ and $\ell(a) = ca + b$ for some constants b, c), this property already ensures that the algorithm is robust

to noise without any further changes. This is because the noise injected to each \mathbf{x}_t merely causes the exact gradient estimate to change to a random vector which is correct in expectation: If we assume ℓ is a classification loss, then

$$\mathbb{E}[\ell'(y_t \langle \mathbf{w}_t, \Psi(\tilde{\mathbf{x}}_t) \rangle) \Psi(\tilde{\mathbf{x}}_t)] = \mathbb{E}[c\tilde{\mathbf{x}}_t] = \mathbf{x}_t.$$

On the other hand, when we use nonlinear kernels and nonlinear loss functions, using standard online gradient descent leads to systematic and unknown biases (since the noise distribution is unknown), which prevents the method from working properly. To deal with this problem, we now turn to describe a technique for estimating expressions such as $\ell'(y_t \langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle)$ in an unbiased manner. In Subsection 3.3, we discuss how $\Psi(\mathbf{x}_t)$ can be estimated in an unbiased manner.

3.2 Unbiased Estimators for Non-Linear Functions

Suppose that we are given access to independent copies of a real random variable X , with expectation $\mathbb{E}[X]$, and some real function f , and we wish to construct an unbiased estimate of $f(\mathbb{E}[X])$. If f is a linear function, then this is easy: just sample x from X , and return $f(x)$. By linearity, $\mathbb{E}[f(X)] = f(\mathbb{E}[X])$ and we are done. The problem becomes less trivial when f is a general, non-linear function, since usually $\mathbb{E}[f(X)] \neq f(\mathbb{E}[X])$. In fact, when X takes finitely many values and f is not a polynomial function, one can prove that no unbiased estimator can exist (see [14], Proposition 8 and its proof). Nevertheless, we show how in many cases one can construct an unbiased estimator of $f(\mathbb{E}[X])$, including cases covered by the impossibility result. There is no contradiction, because we do not construct a “standard” estimator. Usually, an estimator is a function from a given sample to the range of the parameter we wish to estimate. An implicit assumption is that the size of the sample given to it is fixed, and this is also a crucial ingredient in the impossibility result. We circumvent this by constructing an estimator based on a random number of samples.

Here is the key idea: suppose $f : \mathbb{R} \rightarrow \mathbb{R}$ is any function continuous on a bounded interval. It is well known that one can construct a sequence of polynomials $(Q_n(\cdot))_{n=1}^\infty$, where $Q_n(\cdot)$ is a polynomial of degree n , which converges uniformly to f on the interval. If $Q_n(x) = \sum_{i=0}^n \gamma_{n,i} x^i$, let $Q'_n(x_1, \dots, x_n) = \sum_{i=0}^n \gamma_{n,i} \prod_{j=1}^i x_j$. Now, consider the estimator which draws a positive integer N according to some distribution $\mathbb{P}(N = n) = p_n$, samples X for N times to get x_1, x_2, \dots, x_N , and returns $\frac{1}{p_N} (Q'_N(x_1, \dots, x_N) - Q'_{N-1}(x_1, \dots, x_{N-1}))$, where we assume $Q'_0 = 0$. The expected value of this estimator is equal to:

$$\begin{aligned} & \mathbb{E}_{N, x_1, \dots, x_N} \left[\frac{1}{p_N} (Q'_N(x_1, \dots, x_N) - Q'_{N-1}(x_1, \dots, x_{N-1})) \right] \\ &= \sum_{n=1}^{\infty} \frac{p_n}{p_n} \mathbb{E}_{x_1, \dots, x_n} [Q'_n(x_1, \dots, x_n) - Q'_{n-1}(x_1, \dots, x_{n-1})] \\ &= \sum_{n=1}^{\infty} (Q_n(\mathbb{E}[X]) - Q_{n-1}(\mathbb{E}[X])) = f(\mathbb{E}[X]). \end{aligned}$$

Thus, we have an unbiased estimator of $f(\mathbb{E}[X])$.

This technique appeared in a rather obscure early 1960’s paper [16] from sequential estimation theory, and appears to be little known, particularly outside the sequential estimation community. However, we believe this technique is interesting, and expect it to have useful applications for other problems as well.

While this may seem at first like a very general result, the variance of this estimator must be bounded for it to be useful. Unfortunately, this is not true for general continuous functions. More precisely, let N be distributed according to p_n , and let θ be the value returned by the estimator. In [2], it is shown that if X is a Bernoulli random variable, and if $\mathbb{E}[\theta N^k] < \infty$ for some integer $k \geq 1$, then f must be k times continuously differentiable. Since $\mathbb{E}[\theta N^k] \leq (\mathbb{E}[\theta^2] + \mathbb{E}[N^{2k}])/2$, this means that functions f which yield an estimator with finite variance, while using a number of queries with bounded variance, must be continuously differentiable. Moreover, in case we desire the number of queries to be essentially constant (i.e. choose a distribution for N with exponentially decaying tails), we must have $\mathbb{E}[N^k] < \infty$ for all k , which means that f should be infinitely differentiable (in fact, in [2] it is conjectured that f must be analytic in such cases).

Thus, we focus in this paper on functions f which are analytic, i.e., they can be written as $f(x) = \sum_{i=0}^{\infty} \gamma_i x^i$ for appropriate constants $\gamma_0, \gamma_1, \dots$. In that case, Q_n can simply be the truncated Taylor expansion of f to order n , i.e., $Q_n = \sum_{i=0}^n \gamma_i x^i$. Moreover, we can pick $p_n \propto 1/p^n$ for any $p > 1$. So the estimator becomes the following: we sample a nonnegative integer N according

to $\mathbb{P}(N = n) = (p - 1)/p^{n+1}$, sample X independently N times to get x_1, x_2, \dots, x_N , and return $\theta = \gamma_N \frac{p^{N+1}}{p-1} x_1 x_2 \cdots x_N$ where we set $\theta = \frac{p}{p-1} \gamma_0$ if $N = 0$.¹ We have the following:

Lemma 1. *For the above estimator, it holds that $\mathbb{E}[\theta] = f(\mathbb{E}[X])$. The expected number of samples used by the estimator is $1/(p - 1)$, and the probability of it being at least z is p^{-z} . Moreover, if we assume that $f_+(x) = \sum_{n=0}^{\infty} |\gamma_n| x^n$ exists for any x in the domain of interest, then*

$$\mathbb{E}[\theta^2] \leq \frac{p}{p-1} f_+^2 \left(\sqrt{p\mathbb{E}[X^2]} \right).$$

Proof. The fact that $\mathbb{E}[\theta] = f(\mathbb{E}[X])$ follows from the discussion above. The results about the number of samples follow directly from properties of the geometric distribution. As for the second moment, $\mathbb{E}[\theta^2]$ equals

$$\begin{aligned} \mathbb{E}_{N, x_1, \dots, x_N} \left[\gamma_N^2 \frac{p^{2(N+1)}}{(p-1)^2} x_1^2 x_2^2 \cdots x_N^2 \right] &= \sum_{n=0}^{\infty} \frac{(p-1)p^{2(n+1)}}{(p-1)^2 p^{n+1}} \gamma_n^2 \mathbb{E}_{x_1, \dots, x_n} [x_1^2 x_2^2 \cdots x_n^2] \\ &= \frac{p}{p-1} \sum_{n=0}^{\infty} \gamma_n^2 p^n (\mathbb{E}[X^2])^n = \frac{p}{p-1} \sum_{n=0}^{\infty} \left(|\gamma_n| \left(\sqrt{p\mathbb{E}[X^2]} \right)^n \right)^2 \\ &\leq \frac{p}{p-1} \left(\sum_{n=0}^{\infty} |\gamma_n| \left(\sqrt{p\mathbb{E}[X^2]} \right)^n \right)^2 = \frac{p}{p-1} f_+^2 \left(\sqrt{p\mathbb{E}[X^2]} \right). \end{aligned}$$

□

The parameter p provides a *tradeoff* between the variance of the estimator and the number of samples needed: the larger is p , the less samples do we need, but the estimator has more variance. In any case, the sample size distribution decays exponentially fast, so the sample size is essentially bounded.

It should be emphasized that the estimator associated with Lemma 1 is tailored for generality, and is suboptimal in some cases. For example, if f is a polynomial function, then $\gamma_n = 0$ for sufficiently large n , and there is no reason to sample N from a distribution supported on all nonnegative integers - it just increases the variance. Nevertheless, in order to keep the presentation unified and general, we will always use this type of estimator. If needed, the estimator can always be optimized for specific cases.

We also note that this technique can be improved in various directions, if more is known about the distribution of X . For instance, if we have some estimate of the expectation and variance of X , then we can perform a Taylor expansion around the estimated $\mathbb{E}[X]$ rather than 0, and tune the probability distribution of N to be different than the one we used above. These modifications can allow us to make the variance of the estimator arbitrarily small, if the variance of X is small enough. Moreover, one can take polynomial approximations to f which are perhaps better than truncated Taylor expansions. In this paper, for simplicity, we will ignore these potential improvements.

Finally, we note that a related result in [2] implies that it is impossible to estimate $f(\mathbb{E}[X])$ in an unbiased manner when f is discontinuous, even if we allow a number of queries and estimator values which are infinite in expectation. Therefore, since the derivative of the hinge loss is not continuous, estimating in an unbiased manner the gradient of the hinge loss with arbitrary noise appears to be impossible. Thus, if online learning with noise and hinge loss is at all feasible, a rather different approach than ours will need to be taken.

3.3 Unbiasing Noise in the RKHS

The third component of our approach involves the unbiased estimation of $\Psi(\mathbf{x}_t)$, when we only have unbiased noisy copies of \mathbf{x}_t . Here again, we have a non-trivial problem, because the feature mapping Ψ is usually highly non-linear, so $\mathbb{E}[\Psi(\tilde{\mathbf{x}}_t)] \neq \Psi(\mathbb{E}[\tilde{\mathbf{x}}_t])$ in general. Moreover, Ψ is not a scalar function, so the technique of Subsection 3.2 will not work as-is.

To tackle this problem, we construct an explicit feature mapping, which needs to be tailored to the kernel we want to use. To give a very simple example, suppose we use the homogeneous 2nd-degree polynomial kernel, $k(\mathbf{r}, \mathbf{s}) = \langle \mathbf{r}, \mathbf{s} \rangle^2$. It is not hard to verify that the function $\Psi : \mathbb{R}^d \mapsto \mathbb{R}^{d^2}$,

¹Admittedly, the event $N = 0$ should receive zero probability, as it amounts to “skipping” the sampling altogether. However, setting $\mathbb{P}(N = 0) = 0$ appears to improve the bound in this paper only in the smaller order terms, while making the analysis in the paper more complicated.

defined via $\Psi(\mathbf{x}) = (x_1x_1, x_1x_2, \dots, x_dx_d)$, is an explicit feature mapping for this kernel. Now, if we query two independent noisy copies $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}'$ of \mathbf{x} , we have that the expectation of the random vector $(\tilde{x}_1\tilde{x}'_1, \tilde{x}_1\tilde{x}'_2, \dots, \tilde{x}_d\tilde{x}'_d)$ is nothing more than $\Psi(\mathbf{x})$. Thus, we can construct unbiased estimates of $\Psi(\mathbf{x})$ in the RKHS. Of course, this example pertains to a very simple RKHS with a finite dimensional representation. By a randomization trick somewhat similar to the one in Subsection 3.2, we can adapt this approach to infinite dimensional RKHS as well. In a nutshell, we represent $\Psi(\mathbf{x})$ as an infinite-dimensional vector, and its noisy unbiased estimate is a vector which is non-zero on only finitely many entries, using finitely many noisy queries. Moreover, inner products between these estimates can be done efficiently, allowing us to implement the learning algorithms, and use the resulting predictor on test instances.

4 Main Results

4.1 Algorithm

We present our algorithmic approach in a modular form. We start by introducing the main algorithm, which contains several subroutines. Then we prove our two main results, which bound the regret of the algorithm, the number of queries to the oracle, and the running time for two types of kernels: dot product and Gaussian (our results can be extended to other kernel types as well). In itself, the algorithm is nothing more than a standard online gradient descent algorithm with a standard $O(\sqrt{T})$ regret bound. Thus, most of the proofs are devoted to a detailed discussion of how the subroutines are implemented (including explicit pseudo-code). In this section, we just describe one subroutine, based on the techniques discussed in Sec. 3. The other subroutines require a more detailed and technical discussion, and thus their implementation is described as part of the proofs in Sec. 5. In any case, the intuition behind the implementations and the techniques used are described in Sec. 3.

For simplicity, we will focus on a finite-horizon setting, where the number of online rounds T is fixed and known to the learner. The algorithm can easily be modified to deal with the infinite horizon setting, where the learner needs to achieve sub-linear regret for all T simultaneously. Also, for the remainder of this subsection, we assume for simplicity that ℓ is a classification loss, namely can be written as a function of $\ell(y\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle)$. It is not hard to adapt the results below to the case where ℓ is a regression loss (where ℓ is a function of $\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle - y$).

We note that at each round, the algorithm below constructs an object which we denote as $\tilde{\Psi}(\mathbf{x}_t)$. This object has two interpretations here: formally, it is an element of a reproducing kernel Hilbert space (RKHS) corresponding to the kernel we use, and is equal in expectation to $\Psi(\mathbf{x}_t)$. However, in terms of implementation, it is simply a data structure consisting of a finite set of vectors from \mathbb{R}^d . Thus, it can be efficiently stored in memory and handled even for infinite-dimensional RKHS.

Algorithm 1 Kernel Learning Algorithm with Noisy Input

Parameters: Learning rate $\eta > 0$, number of rounds T , sample parameter $p > 1$.
Initialize:
 $\alpha_i = 0$ for all $i = 1, \dots, T$.
 $\tilde{\Psi}(\mathbf{x}_i)$ for all $i = 1, \dots, T$
// $\tilde{\Psi}(\mathbf{x}_i)$ is a data structure which can store a variable number of vectors in \mathbb{R}^d
For $t = 1 \dots T$
 Define $\mathbf{w}_t = \sum_{i=1}^{t-1} \alpha_i \tilde{\Psi}(\mathbf{x}_i)$
 Receive A_t, y_t // The oracle A_t provides noisy estimates of \mathbf{x}_t
 Let $\tilde{\Psi}(\mathbf{x}_t) := \text{Map_Estimate}(A_t, p)$ // Get unbiased estimate of $\Psi(\mathbf{x}_t)$ in the RKHS
 Let $\tilde{g}_t := \text{Grad_Length_Estimate}(A_t, y_t, p)$ // Get unbiased estimate of $\ell'(y_t\langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle)$
 Let $\alpha_t := -\tilde{g}_t\eta/\sqrt{T}$ // Perform gradient step
 Let $\tilde{n}_t := \sum_{i=1}^t \sum_{j=1}^t \alpha_{t,i}\alpha_{t,j} \text{Prod}(\tilde{\Psi}(\mathbf{x}_i), \tilde{\Psi}(\mathbf{x}_j))$
 // Compute squared norm, where $\text{Prod}(\tilde{\Psi}(\mathbf{x}_i), \tilde{\Psi}(\mathbf{x}_j))$ returns $\langle \tilde{\Psi}(\mathbf{x}_i), \tilde{\Psi}(\mathbf{x}_j) \rangle$
 If $\tilde{n}_t > B_w$ // If norm squared is larger than B_w , then project
 Let $\alpha_i := \alpha_i \frac{\sqrt{B_w}}{\tilde{n}_t}$ for all $i = 1, \dots, t$

Like $\tilde{\Psi}(\mathbf{x}_t)$, \mathbf{w}_{t+1} has also two interpretations: formally, it is an element in the RKHS, as defined in the pseudocode. In terms of implementation, it is defined via the data structures $\tilde{\Psi}(\mathbf{x}_1), \dots, \tilde{\Psi}(\mathbf{x}_t)$ and the values of $\alpha_1, \dots, \alpha_t$ at round t . To apply this hypothesis on a given instance \mathbf{x} , we compute

$\sum_{i=1}^t \alpha_{t,i} \text{Prod}(\tilde{\Psi}(\mathbf{x}_i), \mathbf{x}')$, where $\text{Prod}(\tilde{\Psi}(\mathbf{x}_i), \mathbf{x}')$ is a subroutine which returns $\langle \tilde{\Psi}(\mathbf{x}_i), \Psi(\mathbf{x}') \rangle$ (a pseudocode is provided as part of the proofs later on).

We now turn to the main results pertaining to the algorithm. The first result shows what regret bound is achievable by the algorithm for any dot-product kernel, as well as characterize the number of oracle queries per instance, and the overall running time of the algorithm.

Theorem 1. *Assume that the loss function ℓ has an analytic derivative $\ell'(a) = \sum_{n=0}^{\infty} \gamma_n a^n$ for all a in its domain, and let $\ell'_+(a) = \sum_{n=0}^{\infty} |\gamma_n| a^n$ (assuming it exists). Assume also that the kernel $k(\mathbf{x}, \mathbf{x}')$ can be written as $Q(\langle \mathbf{x}, \mathbf{x}' \rangle)$ for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$. Finally, assume that $\mathbb{E}[\|\tilde{\mathbf{x}}_t\|^2] \leq B_{\tilde{\mathbf{x}}}$ for any $\tilde{\mathbf{x}}_t$ returned by the oracle at round t , for all $t = 1, \dots, T$. Then, for all $B_{\mathbf{w}} > 0$ and $p > 1$, it is possible to implement the subroutines of Algorithm 1 such that:*

- The expected number of queries to each oracle A_t is $\frac{p}{(p-1)^2}$.
- The expected running time of the algorithm is $O\left(T^3 \left(1 + \frac{dp}{(p-1)^2}\right)\right)$.
- If we run Algorithm 1 with $\eta = B_{\mathbf{w}} / \sqrt{u} \ell'_+(\sqrt{(p-1)u})$, where $u = B_{\mathbf{w}} \left(\frac{p}{p-1}\right)^2 Q(pB_{\tilde{\mathbf{x}}})$, then

$$\mathbb{E} \left[\sum_{t=1}^T \ell(y_t \langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle) - \min_{\mathbf{w}: \|\mathbf{w}\|^2 \leq B_{\mathbf{w}}} \sum_{t=1}^T \ell(y_t \langle \mathbf{w}, \Psi(\mathbf{x}_t) \rangle) \right] \leq \ell'_+(\sqrt{(p-1)u}) \sqrt{uT}.$$

The expectations are with respect to the randomness of the oracles and the algorithm throughout its run.

We note that the distribution of the number of oracle queries can be specified explicitly, and it decays very rapidly - see the proof for details. Also, for simplicity, we only bound the expected regret in the theorem above. If the noise is bounded almost surely or with sub-Gaussian tails (rather than just bounded variance), then it is possible to obtain similar guarantees with high probability, by relying on Azuma's inequality or variants thereof (see for example [4]).

We now turn to the case of Gaussian kernels.

Theorem 2. *Assume that the loss function ℓ has an analytic derivative $\ell'(a) = \sum_{n=0}^{\infty} \gamma_n a^n$ for all a in its domain, and let $\ell'_+(a) = \sum_{n=0}^{\infty} |\gamma_n| a^n$ (assuming it exists). Assume that the kernel $k(\mathbf{x}, \mathbf{x}')$ is defined as $\exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / \sigma^2)$. Finally, assume that $\mathbb{E}[\|\tilde{\mathbf{x}}_t\|^2] \leq B_{\tilde{\mathbf{x}}}$ for any $\tilde{\mathbf{x}}_t$ returned by the oracle at round t , for all $t = 1, \dots, T$. Then for all $B_{\mathbf{w}} > 0$ and $p > 1$ it is possible to implement the subroutines of Algorithm 1 such that*

- The expected number of queries to each oracle A_t is $\frac{3p}{(p-1)^2}$.
- The expected running time of the algorithm is $O\left(T^3 \left(1 + \frac{dp}{(p-1)^2}\right)\right)$.
- If we run Algorithm 1 with $\eta = B_{\mathbf{w}} / \sqrt{u} \ell'_+(\sqrt{(p-1)u})$, where

$$u = B_{\mathbf{w}} \left(\frac{p}{p-1}\right)^3 \exp\left(\frac{\sqrt{p}B_{\tilde{\mathbf{x}}} + 2p\sqrt{B_{\tilde{\mathbf{x}}}}}{\sigma^2}\right)$$

then

$$\mathbb{E} \left[\sum_{t=1}^T \ell(y_t \langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle) - \min_{\mathbf{w}: \|\mathbf{w}\|^2 \leq B_{\mathbf{w}}} \sum_{t=1}^T \ell(y_t \langle \mathbf{w}, \Psi(\mathbf{x}_t) \rangle) \right] \leq \ell'_+(\sqrt{(p-1)u}) \sqrt{uT}.$$

The expectations are with respect to the randomness of the oracles and the algorithm throughout its run.

As in Thm. 1, note that the number of oracle queries has a fast decaying distribution. Also, note that with Gaussian kernels, σ^2 is usually chosen to be on the order of the example's squared norms. Thus, if the noise added to the examples is proportional to their original norm, we can assume that $B_{\tilde{\mathbf{x}}}/\sigma^2 = O(1)$, and thus u which appears in the bound is also bounded by a constant.

As previously mentioned, most of the subroutines are described in the proofs section, as part of the proof of Thm. 1. Here, we only show how to implement `Grad.Length.Estimate` subroutine, which returns the gradient length estimate \tilde{g}_t . The idea is based on the technique described in

Subsection 3.2. We prove that \tilde{g}_t is an unbiased estimate of $\ell'(y_t \langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle)$, and bound $\mathbb{E}[\tilde{g}_t^2]$. As discussed earlier, we assume that $\ell'(\cdot)$ is analytic and can be written as $\ell'(a) = \sum_{n=0}^{\infty} \gamma_n a^n$.

Subroutine 1 `Grad_Length_Estimate`(A_t, y_t, p)

Sample nonnegative integer n according to $\mathbb{P}(n) = (p-1)/p^{n+1}$

For $j = 1, \dots, n$

Let $\tilde{\Psi}(\mathbf{x}_t)_j := \text{Map_Estimate}(A_t)$ // Get unbiased estimate of $\Psi(\mathbf{x}_t)$ in the RKHS

Return $\tilde{g}_t := y_t \gamma_n \frac{p^{n+1}}{p-1} \prod_{j=1}^n \left(\sum_{i=1}^{t-1} \alpha_{t-1,i} \text{Prod}(\tilde{\Psi}(\mathbf{x}_i), \tilde{\Psi}(\mathbf{x}_t)_j) \right)$

Lemma 2. Assume that $\mathbb{E}[\tilde{\Psi}(\mathbf{x}_t)] = \Psi(\mathbf{x}_t)$, and that `Prod`($\tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}')$) returns $\langle \tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}') \rangle$ for all \mathbf{x}, \mathbf{x}' . Then for any given $\mathbf{w}_t = \alpha_{t-1,1} \tilde{\Psi}(\mathbf{x}_1) + \dots + \alpha_{t-1,t-1} \tilde{\Psi}(\mathbf{x}_{t-1})$ it holds that

$$\mathbb{E}_t[\tilde{g}_t] = y_t \ell'(y_t \langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle) \quad \text{and} \quad \mathbb{E}_t[\tilde{g}_t^2] \leq \frac{p}{p-1} \ell'_+ \left(\sqrt{p B_{\mathbf{w}} B_{\tilde{\Psi}(\mathbf{x})}} \right)^2$$

where the expectation is with respect to the randomness of Subroutine 1, and $\ell'_+(a) = \sum_{n=0}^{\infty} |\gamma_n| a^n$.

Proof. The result follows from Lemma 1, where \tilde{g}_t corresponds to the estimator θ , the function f corresponds to ℓ' , and the random variable X corresponds to $\langle \mathbf{w}_t, \tilde{\Psi}(\mathbf{x}_t) \rangle$ (where $\tilde{\Psi}(\mathbf{x}_t)$ is random and \mathbf{w}_t is held fixed). The term $\mathbb{E}[X^2]$ in Lemma 1 can be upper bounded as

$$\mathbb{E}_t \left[(\langle \mathbf{w}_t, \tilde{\Psi}(\mathbf{x}_t) \rangle)^2 \right] \leq \|\mathbf{w}_t\|^2 \mathbb{E}_t \left[\|\tilde{\Psi}(\mathbf{x}_t)\|^2 \right] \leq B_{\mathbf{w}} B_{\tilde{\Psi}(\mathbf{x})} .$$

□

4.2 Loss Function Examples

Theorems 1 and 2 both deal with generic loss functions ℓ whose derivative can be written as $\sum_{n=0}^{\infty} \gamma_n a^n$, and the regret bounds involve the functions $\ell'_+(a) = \sum_{n=0}^{\infty} |\gamma_n| a^n$. Below, we present a few examples of loss functions and their corresponding ℓ'_+ . As mentioned earlier, while the theorems in the previous subsection are in terms of classification losses (i.e., ℓ is a function of $y \langle \mathbf{w}, \Psi(\mathbf{x}) \rangle$), virtually identical results can be proven for regression losses (i.e., ℓ is a function of $\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle - y$), so we will give examples from both families. Working out the first two examples is straightforward. The proofs of the other two appear in Sec. 5. The loss functions are illustrated graphically in Fig. 1.

Example 1. For the squared loss function, $\ell(\langle \mathbf{w}, \mathbf{x} \rangle, y) = (\langle \mathbf{w}, \mathbf{x} \rangle - y)^2$, we have $\ell'_+(\sqrt{(p-1)u}) = 2\sqrt{(p-1)u}$.

Example 2. For the exponential loss function, $\ell(\langle \mathbf{w}, \mathbf{x} \rangle, y) = e^{y \langle \mathbf{w}, \mathbf{x} \rangle}$, we have $\ell'_+(\sqrt{(p-1)u}) = e^{\sqrt{(p-1)u}}$.

Example 3. Consider a “smoothed” absolute loss function $\ell_\sigma(\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle - y)$, defined as an antiderivative of $\text{Erf}(sa)$ for some $s > 0$ (see proof for exact analytic form). Then we have that $\ell'_+(\sqrt{(p-1)u}) \leq \frac{1}{2} + \frac{1}{s\sqrt{\pi(p-1)u}} (e^{s^2(p-1)u} - 1)$.

Example 4. Consider a “smoothed” hinge loss $\ell(y \langle \mathbf{w}, \Psi(\mathbf{x}) \rangle)$, defined as an antiderivative of $(\text{Erf}(s(a-1)) - 1)/2$ for some $s > 0$ (see proof for exact analytic form). Then we have that $\ell'_+(\sqrt{(p-1)u}) \leq \frac{2}{s\sqrt{\pi(p-1)u}} (e^{s^2(p-1)u-1})$.

For any s , the loss function in the last two examples are convex, and respectively approximate the absolute loss $|\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle - y|$ and the hinge loss $\max\{0, 1 - y \langle \mathbf{w}, \Psi(\mathbf{x}) \rangle\}$ arbitrarily well for large enough s . Fig. 1 shows these loss functions graphically for $s = 1$. Note that s need not be large in order to get a good approximation. Also, we note that both the loss itself and its gradient are computationally easy to evaluate.

Finally, we remind the reader that as discussed in Subsection 3.2, performing an unbiased estimate of the gradient for non-differentiable losses directly (such as the hinge loss or absolute loss) appears to be impossible in general. On the flip side, if one is willing to use a random number of queries with polynomial rather than exponential tails, then one can achieve much better sample complexity results, by focusing on loss functions (or approximations thereof) which are only differentiable to a bounded order, rather than fully analytic. This again demonstrates the tradeoff between the sample size and the amount of information that needs to be gathered on each training example.

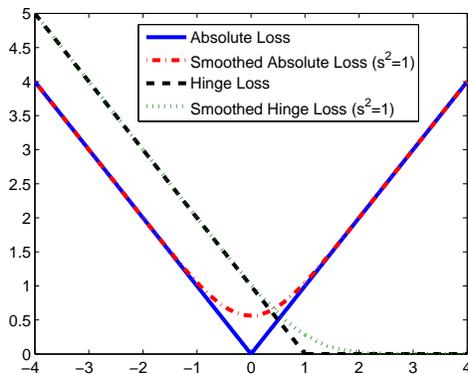


Figure 1: Absolute loss, hinge loss, and smooth approximations

4.3 One Noisy Copy is Not Enough

The previous results might lead one to wonder whether it is really necessary to query the same instance more than once. In some applications this is inconvenient, and one would prefer a method which works when just a single noisy copy of each instance is made available. In this subsection we show that, unfortunately, such a method cannot be found. Specifically, we prove that under very mild assumptions, no method can achieve sub-linear regret when it has access to just a single noisy copy of each instance. On the other hand, for the case of squared loss and linear kernels, our techniques can be adapted to work with exactly two noisy copies of each instance,² so without further assumptions, the lower bound that we prove here is indeed tight. For simplicity, we prove the result for linear kernels (i.e., where $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$). It is an interesting open problem to show improved lower bounds when nonlinear kernels are used. We also note that the result crucially relies on the learner not knowing the noise distribution, and we leave to future work the investigation of what happens when this assumption is relaxed.

Theorem 3. *Let \mathcal{W} be a compact convex subset of \mathbb{R}^d , and let $\ell(\cdot, 1) : \mathbb{R} \mapsto \mathbb{R}$ satisfies the following: (1) it is bounded from below; (2) it is differentiable at 0 with $\ell'(0, 1) < 0$. For any learning algorithm which selects hypotheses from \mathcal{W} and is allowed access to a single noisy copy of the instance at each round t , there exists a strategy for the adversary such that the sequence $\mathbf{w}_1, \mathbf{w}_2, \dots$ of predictors output by the algorithm satisfies*

$$\limsup_{T \rightarrow \infty} \max_{\mathbf{w} \in \mathcal{W}} \frac{1}{T} \sum_{t=1}^T \left(\ell(\langle \mathbf{w}_t, \mathbf{x}_t \rangle, y_t) - \ell(\langle \mathbf{w}, \mathbf{x}_t \rangle, y_t) \right) > 0$$

with probability 1 with respect to the randomness of the oracles.

Note that condition (1) is satisfied by virtually any loss function other than the linear loss, while condition (2) is satisfied by most regression losses, and by all *classification calibrated losses*, which include all reasonable losses for classification (see [13]). The most obvious example where the conditions are not satisfied is when $\ell(\cdot, 1)$ is a linear function. This is not surprising, because when $\ell(\cdot, 1)$ is linear, the learner is always robust to noise (see the discussion at Sec. 3).

The intuition of the proof is very simple: the adversary chooses beforehand whether the examples are drawn i.i.d. from a distribution \mathcal{D} , and then perturbed by noise, or drawn i.i.d. from some other distribution \mathcal{D}' without adding noise. The distributions $\mathcal{D}, \mathcal{D}'$ and the noise are designed so that the examples observed by the learner are distributed in the same way irrespective to which of the two sampling strategies the adversary chooses. Therefore, it is impossible for the learner accessing a single copy of each instance to be statistically consistent with respect to both distributions simultaneously. As a result, the adversary can always choose a distribution on which the algorithm will be inconsistent, leading to constant regret. The full proof is presented in Section 5.3.

²In a nutshell, for squared loss and linear kernels, we just need to estimate $2(\langle \mathbf{w}_t, \mathbf{x}_t \rangle - y_t)\mathbf{x}_t$ in an unbiased manner at each round t . This can be done by computing $2(\langle \mathbf{w}_t, \tilde{\mathbf{x}}_t \rangle - y_t)\tilde{\mathbf{x}}'_t$, where $\tilde{\mathbf{x}}_t, \tilde{\mathbf{x}}'_t$ are two noisy copies of \mathbf{x}_t .

5 Proofs

Due to the lack of space, some of the proofs are given in the [7].

5.1 Preliminary Result

To prove Thm. 1 and Thm. 2, we need a theorem which basically states that if all subroutines in algorithm 1 behave as they should, then one can achieve an $O(\sqrt{T})$ regret bound. This is provided in the following theorem, which is an adaptation of a standard result of online convex optimization (see, e.g., [18]). The proof is given in [7].

Theorem 4. *Assume the following conditions hold with respect to Algorithm 1:*

1. For all t , $\tilde{\Psi}(\mathbf{x}_t)$ and \tilde{g}_t are independent of each other (as random variables induced by the randomness of Algorithm 1) as well as independent of any $\tilde{\Psi}(\mathbf{x}_i)$ and \tilde{g}_i for $i < t$.
2. For all t , $\mathbb{E}[\tilde{\Psi}(\mathbf{x}_t)] = \Psi(\mathbf{x}_t)$, and there exists a constant $B_{\tilde{\Psi}} > 0$ such that $\mathbb{E}[\|\tilde{\Psi}(\mathbf{x}_t)\|^2] \leq B_{\tilde{\Psi}}$.
3. For all t , $\mathbb{E}[\tilde{g}_t] = y_t \ell'(y_t \langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle)$, and there exists a constant $B_{\tilde{g}} > 0$ such that $\mathbb{E}[\tilde{g}_t^2] \leq B_{\tilde{g}}$.
4. For any pair of instances \mathbf{x}, \mathbf{x}' , $\text{Prod}(\tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}')) = \langle \tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}') \rangle$.

Then if Algorithm 1 is run with $\eta = \sqrt{\frac{B_{\mathbf{w}}}{B_{\tilde{g}} B_{\tilde{\Psi}}}}$, the following inequality holds

$$\mathbb{E} \left[\sum_{t=1}^T \ell(y_t \langle \mathbf{w}_t, \Psi(\mathbf{x}_t) \rangle) - \min_{\mathbf{w}: \|\mathbf{w}\|^2 \leq B_{\mathbf{w}}} \sum_{t=1}^T \ell(y_t \langle \mathbf{w}, \Psi(\mathbf{x}_t) \rangle) \right] \leq \sqrt{B_{\mathbf{w}} B_{\tilde{g}} B_{\tilde{\Psi}} T}.$$

where the expectation is with respect to the randomness of the oracles and the algorithm throughout its run.

5.2 Proof of Thm. 1

In this subsection, we present the proof of Thm. 1. We first show how to implement the subroutines of Algorithm 1, and prove the relevant results on their behavior. Then, we prove the theorem itself.

It is known that for $k(\cdot, \cdot) = Q(\langle \mathbf{x}, \mathbf{x}' \rangle)$ to be a valid kernel, it is necessary that $Q(\langle \mathbf{x}, \mathbf{x}' \rangle)$ can be written as a Taylor expansion $\sum_{n=0}^{\infty} \beta_n (\langle \mathbf{x}, \mathbf{x}' \rangle)^n$, where $\beta_n \geq 0$ (see theorem 4.19 in [15]). This makes these types of kernels amenable to our techniques.

We start by constructing an explicit feature mapping $\Psi(\cdot)$ corresponding to the RKHS induced by our kernel. For any \mathbf{x}, \mathbf{x}' , we have that

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \sum_{n=0}^{\infty} \beta_n (\langle \mathbf{x}, \mathbf{x}' \rangle)^n = \sum_{n=0}^{\infty} \beta_n \left(\sum_{i=1}^d x_i x'_i \right)^n \\ &= \sum_{n=0}^{\infty} \beta_n \sum_{k_1=1}^d \cdots \sum_{k_n=1}^d x_{k_1} x_{k_2} \cdots x_{k_n} x'_{k_1} x'_{k_2} \cdots x'_{k_n} \\ &= \sum_{n=0}^{\infty} \sum_{k_1=1}^d \cdots \sum_{k_n=1}^d \left(\sqrt{\beta_n} x_{k_1} x_{k_2} \cdots x_{k_n} \right) \left(\sqrt{\beta_n} x'_{k_1} x'_{k_2} \cdots x'_{k_n} \right). \end{aligned}$$

This suggests the following feature representation: for any \mathbf{x} , $\Psi(\mathbf{x})$ returns an infinite-dimensional vector, indexed by n and $k_1, \dots, k_n \in \{1, \dots, d\}$, with the entry corresponding to n, k_1, \dots, k_n being $\sqrt{\beta_n} x_{k_1} \cdots x_{k_n}$. The dot product between $\Psi(\mathbf{x})$ and $\Psi(\mathbf{x}')$ is similar to a standard dot product between two vectors, and by the derivation above equals $k(\mathbf{x}, \mathbf{x}')$ as required.

We now use a slightly more elaborate variant of our unbiased estimate technique, to derive an unbiased estimate of $\Psi(\mathbf{x})$. First, we sample N according to $\mathbb{P}(N = n) = (p-1)/p^{n+1}$. Then, we query the oracle for \mathbf{x} for N times to get $\tilde{\mathbf{x}}^{(1)}, \dots, \tilde{\mathbf{x}}^{(N)}$, and formally define $\tilde{\Psi}(\mathbf{x})$ as

$$\tilde{\Psi}(\mathbf{x}) = \sqrt{\beta_n} \frac{p^{n+1}}{p-1} \sum_{k_1=1}^d \cdots \sum_{k_n=1}^d \tilde{x}_{k_1}^{(1)} \cdots \tilde{x}_{k_n}^{(n)} \mathbf{e}_{n, k_1, \dots, k_n} \quad (2)$$

where $\mathbf{e}_{n, k_1, \dots, k_n}$ represents the unit vector in the direction indexed by n, k_1, \dots, k_n as explained above. Since the oracle queries are i.i.d., the expectation of this expression is

$$\sum_{n=0}^{\infty} \frac{p-1}{p^{n+1}} \sqrt{\beta_n} \frac{p^{n+1}}{p-1} \sum_{k_1=1}^d \cdots \sum_{k_n=1}^d \mathbb{E}[\tilde{x}_{k_1}^{(1)} \cdots \tilde{x}_{k_n}^{(n)}] \mathbf{e}_{n, k_1, \dots, k_n} = \sum_{n=0}^{\infty} \sum_{k_1=1}^d \cdots \sum_{k_n=1}^d \sqrt{\beta_n} x_{k_1}^{(1)} \cdots x_{k_n}^{(n)} \mathbf{e}_{n, k_1, \dots, k_n}$$

which is exactly $\Psi(\mathbf{x})$. We formalize the needed properties of $\tilde{\Psi}(\mathbf{x})$ in the following lemma.

Lemma 3. Assuming $\tilde{\Psi}(\mathbf{x})$ is constructed as in the discussion above, it holds that $\mathbb{E}[\tilde{\Psi}(\mathbf{x})] = \Psi(\mathbf{x})$ for any \mathbf{x} . Moreover, if the noisy samples $\tilde{\mathbf{x}}_t$ returned by the oracle A_t satisfy $\mathbb{E}[\|\tilde{\mathbf{x}}_t\|^2] \leq B_{\tilde{\mathbf{x}}}$, then

$$\mathbb{E} \left[\|\tilde{\Psi}(\mathbf{x}_t)\|^2 \right] \leq \frac{p}{p-1} Q(pB_{\tilde{\mathbf{x}}})$$

where we recall that Q defines the kernel by $k(\mathbf{x}, \mathbf{x}') = Q(\langle \mathbf{x}, \mathbf{x}' \rangle)$.

Proof. The first part of the lemma follows from the discussion above. As to the second part, note that by (2),

$$\begin{aligned} \mathbb{E} \left[\|\tilde{\Psi}(\mathbf{x}_t)\|^2 \right] &= \mathbb{E} \left[\beta_n \frac{p^{2n+2}}{(p-1)^2} \sum_{k_1, \dots, k_n=1}^d \left(\tilde{x}_{t,k_1}^{(1)} \cdots \tilde{x}_{t,k_n}^{(N)} \right)^2 \right] = \mathbb{E} \left[\beta_n \frac{p^{2n+2}}{(p-1)^2} \prod_{j=1}^n \|\tilde{\mathbf{x}}_t^{(j)}\|^2 \right] \\ &= \sum_{n=0}^{\infty} \frac{p-1}{p^{n+1}} \beta_n \frac{p^{2n+2}}{(p-1)^2} (\mathbb{E}[\tilde{\mathbf{x}}_t^2])^n = \frac{p}{p-1} \sum_{n=0}^{\infty} \beta_n (p\mathbb{E}[\tilde{\mathbf{x}}_t^2])^n \leq \frac{p}{p-1} \sum_{n=0}^{\infty} \beta_n (pB_{\tilde{\mathbf{x}}})^n = \frac{p}{p-1} Q(pB_{\tilde{\mathbf{x}}}) \end{aligned}$$

where the second-to-last step used the fact that $\beta_n \geq 0$ for all n . \square

Of course, explicitly storing $\tilde{\Psi}(\mathbf{x})$ as defined above is infeasible, since the number of entries is huge. Fortunately, this is not needed: we just need to store $\tilde{\mathbf{x}}_t^{(1)}, \dots, \tilde{\mathbf{x}}_t^{(N)}$. The representation above is used implicitly when we calculate dot products between $\tilde{\Psi}(\mathbf{x})$ and other elements in the RKHS, via the subroutine `Prod`. We note that while N is a random quantity which might be unbounded, its distribution decays exponentially fast, so the number of vectors to store is essentially bounded.

After the discussion above, the pseudocode for `Map_Estimate` below should be self-explanatory.

Subroutine 2 `Map_Estimate`(A_t, p)

Sample nonnegative integer N according to $\mathbb{P}(N = n) = (p-1)/p^{n+1}$

Query A_t for N times to get $\tilde{\mathbf{x}}_t^{(1)}, \dots, \tilde{\mathbf{x}}_t^{(N)}$

Return $\tilde{\mathbf{x}}_t^{(1)}, \dots, \tilde{\mathbf{x}}_t^{(N)}$ as $\tilde{\Psi}(\mathbf{x}_t)$.

We now turn to the subroutine `Prod`, which given two elements in the RKHS, returns their dot product. This subroutine comes in two flavors: either as a procedure defined over $\tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}')$ and returning $\langle \tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}') \rangle$ (Subroutine 3); or as a procedure defined over $\tilde{\Psi}(\mathbf{x}), \mathbf{x}'$ (Subroutine 4, where the second element is an explicitly given vector) and returning $\langle \tilde{\Psi}(\mathbf{x}), \Psi(\mathbf{x}') \rangle$. This second variant of `Prod` is needed when we wish to apply the learned predictor on a new given instance \mathbf{x}' .

Subroutine 3 `Prod`($\tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}')$)

Let $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}$ be the index and vectors comprising $\Psi(\mathbf{x})$

Let $\mathbf{x}'^{(1)}, \dots, \mathbf{x}'^{(n')}$ be the index and vectors comprising $\Psi(\mathbf{x}')$

If $n \neq n'$ return 0, else return $\beta_n \frac{p^{2n+2}}{(p-1)^2} \prod_{j=1}^n \langle \tilde{\mathbf{x}}^{(j)}, \tilde{\mathbf{x}}'^{(j)} \rangle$

Lemma 4. `Prod`($\tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}')$) returns $\langle \tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}') \rangle$.

Proof. Using the formal representation of $\tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}')$ in (2), we have that $\langle \tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}') \rangle$ is 0 whenever $n \neq n'$ (because then these two elements are composed of different unit vectors with respect to an orthogonal basis). Otherwise, we have that

$$\begin{aligned} \langle \tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}') \rangle &= \beta_n \frac{p^{2n+2}}{(p-1)^2} \sum_{k_1, \dots, k_n=1}^d \tilde{x}_{k_1}^{(1)} \cdots \tilde{x}_{k_n}^{(n)} \tilde{x}'_{k_1}{}^{(1)} \cdots \tilde{x}'_{k_n}{}^{(n)} \\ &= \beta_n \frac{p^{2n+2}}{(p-1)^2} \left(\sum_{k_1=1}^d \tilde{x}_{k_1}^{(1)} \tilde{x}'_{k_1}{}^{(1)} \right) \cdots \left(\sum_{k_N=1}^d \tilde{x}_{k_N}^{(n)} \tilde{x}'_{k_N}{}^{(n)} \right) = \beta_n \frac{p^{2n+2}}{(p-1)^2} \prod_{j=1}^n \left(\langle \tilde{\mathbf{x}}^{(j)}, \tilde{\mathbf{x}}'^{(j)} \rangle \right) \end{aligned}$$

which is exactly what the algorithm returns, hence the lemma follows. \square

The pseudocode for calculating the dot product $\langle \tilde{\Psi}(\mathbf{x}), \Psi(\mathbf{x}') \rangle$ (where \mathbf{x}' is known) is very similar, and the proof is essentially the same.

Subroutine 4 $\text{Prod}(\tilde{\Psi}(\mathbf{x}), \mathbf{x}')$

Let $n, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}$ be the index and vectors comprising $\Psi(\mathbf{x})$
 Return $\beta_n \frac{p^{n+1}}{p-1} \prod_{j=1}^n \langle \tilde{\mathbf{x}}^{(j)}, \mathbf{x}' \rangle$

We are now ready to prove Thm. 1. First, regarding the expected number of queries, notice that to run Algorithm 1, we invoke `Map_Estimate` and `Grad_Length_Estimate` once at round t . `Map_Estimate` uses a random number B of queries distributed as $\mathbb{P}(B = n) = (p-1)/p^{n+1}$, and `Grad_Length_Estimate` invokes `Map_Estimate` a random number C of times, distributed as $\mathbb{P}(C = n) = (p-1)/p^{n+1}$. The total number of queries is therefore $\sum_{j=1}^{C+1} B_j$, where B_j for all j are i.i.d. copies of B . The expected value of this expression, using a standard result on the expected value of a sum of a random number of independent random variables, is equal to $(1 + \mathbb{E}[C])\mathbb{E}[B_j]$, or $(1 + \frac{1}{p-1})\frac{1}{p-1} = \frac{p}{(p-1)^2}$.

In terms of running time, we note that the expected running time of `Prod` is $O(1 + \frac{d}{p-1})$, this because it performs N multiplications of inner products, each one with running time $O(d)$, and $\mathbb{E}[N] = \frac{1}{p-1}$. The expected running time of `Map_Estimate` is $O(1 + \frac{1}{p-1})$. The expected running time of `Grad_Length_Estimate` is $O(1 + \frac{1}{p-1}(1 + \frac{1}{p-1}) + T(1 + \frac{d}{p-1}))$, which can be written as $O(\frac{p}{(p-1)^2} + T(1 + \frac{d}{p-1}))$. Since Algorithm 1 at each of T rounds calls `Map_Estimate` once, `Grad_Length_Estimate` once, `Prod` for $O(T^2)$ times, and performs $O(1)$ other operations, we get that the overall runtime is

$$O\left(T\left(1 + \frac{1}{p-1} + \frac{p}{(p-1)^2} + T\left(1 + \frac{d}{p-1}\right) + T^2\left(1 + \frac{d}{p-1}\right)\right)\right).$$

Since $\frac{1}{p-1} \leq \frac{p}{(p-1)^2}$, we can upper bound this by

$$O\left(T\left(1 + \frac{p}{(p-1)^2} + T^2\left(1 + \frac{dp}{(p-1)^2}\right)\right)\right) = O\left(T^3\left(1 + \frac{dp}{(p-1)^2}\right)\right).$$

The regret bound in the theorem follows from Thm. 4, with the expressions for constants following from Lemma 2, Lemma 3, and Lemma 4.

5.3 Proof Sketch of Thm. 3

To prove the theorem, we use a more general result which leads to non-vanishing regret, and then show that under the assumptions of Thm. 3, the result holds. The proof of the result is given in [7].

Theorem 5. *Let \mathcal{W} be a compact convex subset of \mathbb{R}^d and pick any learning algorithm which selects hypotheses from \mathcal{W} and is allowed access to a single noisy copy of the instance at each round t . If there exists a distribution over a compact subset of \mathbb{R}^d such that*

$$\operatorname{argmin}_{\mathbf{w} \in \mathcal{W}} \mathbb{E}[\ell(\langle \mathbf{w}, \mathbf{x} \rangle, 1)] \quad \text{and} \quad \operatorname{argmin}_{\mathbf{w} \in \mathcal{W}} \ell(\langle \mathbf{w}, \mathbb{E}[\mathbf{x}] \rangle, 1) \tag{3}$$

are disjoint, then there exists a strategy for the adversary such that the sequence $\mathbf{w}_1, \mathbf{w}_2, \dots \in \mathcal{W}$ of predictors output by the algorithm satisfies

$$\limsup_{T \rightarrow \infty} \max_{\mathbf{w} \in \mathcal{W}} \frac{1}{T} \sum_{t=1}^T \left(\ell(\langle \mathbf{w}_t, \mathbf{x}_t \rangle, y_t) - \ell(\langle \mathbf{w}, \mathbf{x}_t \rangle, y_t) \right) > 0$$

with probability 1 with respect to the randomness of the oracles.

Another way to phrase this theorem is that the regret cannot vanish, if given examples sampled i.i.d. from a distribution, the learning problem is more complicated than just finding the mean of the data. Indeed, the adversary's strategy we choose later on is simply drawing and presenting examples from such a distribution. Below, we sketch how we use Thm. 5 in order to prove Thm. 3. A full proof is provided in [7].

We construct a very simple one-dimensional distribution, which satisfies the conditions of Thm. 5: it is simply the uniform distribution on $\{3\mathbf{x}, -\mathbf{x}\}$, where \mathbf{x} is the vector $(1, 0, \dots, 0)$. Thus, it is enough to show that

$$\operatorname{argmin}_{w: |w|^2 \leq B_{\mathbf{w}}} \ell(3w, 1) + \ell(-w, 1) \quad \text{and} \quad \operatorname{argmin}_{w: |w|^2 \leq B_{\mathbf{w}}} \ell(w, 1) \quad (4)$$

are disjoint, for some appropriately chosen $B_{\mathbf{w}}$. Assuming the contrary, then under the assumptions on ℓ , we show that the first set in Eq. (4) is inside a bounded ball around the origin, in a way which is independent of $B_{\mathbf{w}}$, no matter how large it is. Thus, if we pick $B_{\mathbf{w}}$ to be large enough, and assume that the two sets in Eq. (4) are not disjoint, then there must be some w such that both $\ell(3w, 1) + \ell(-w, 1)$ and $\ell(w, 1)$ have a subgradient of zero at w . However, this can be shown to contradict the assumptions on ℓ , leading to the desired result.

6 Future Work

There are several interesting research directions worth pursuing in the noisy learning framework introduced here. For instance, doing away with unbiasedness, which could lead to the design of estimators that are applicable to more types of loss functions, for which unbiased estimators may not even exist. Also, it would be interesting to show how additional information one has about the noise distribution can be used to design improved estimates, possibly in association with specific losses or kernels. Another open question is whether our lower bound (Thm. 3) can be improved when nonlinear kernels are used.

References

- [1] J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *COLT*, pages 263–274, 2008.
- [2] S. Bhandari and A. Bose. Existence of unbiased estimators in sequential binomial experiments. *Sankhyā: The Indian Journal of Statistics*, 52(1):127–130, 1990.
- [3] N. Bshouty, J. Jackson, and C. Tamon. Uniform-distribution attribute noise learnability. *Information and Computation*, 187(2):277–290, 2003.
- [4] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, September 2004.
- [5] N. Cesa-Bianchi, E. Dichterman, P. Fischer, E. Shamir, and H. Simon. Sample-efficient strategies for learning in the presence of noise. *Journal of the ACM*, 46(5):684–719, 1999.
- [6] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.
- [7] N. Cesa-Bianchi, S. Shalev-Shwartz, and O. Shamir. Online learning of noisy data with kernels. *Technical Report*, available at arXiv:1005.2996.
- [8] A. Flaxman, A. Tauman Kalai, and H. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of SODA*, pages 385–394, 2005.
- [9] S. Goldman and R. Sloan. Can pac learning algorithms tolerate random attribute noise? *Algorithmica*, 14(1):70–84, 1995.
- [10] M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.
- [11] N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow. In *Proceedings of COLT*, pages 147–156, 1991.
- [12] D. Nettleton, A. Orriols-Puig, and A. Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial Intelligence Review*, 2010.
- [13] M. Jordan P. Bartlett and J. McAuliffe. Convexity, classification and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, March 2006.
- [14] L. Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15(6):1191–1253, 2003.
- [15] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.
- [16] R. Singh. Existence of unbiased estimates. *Sankhyā: The Indian Journal of Statistics*, 26(1):93–96, 1964.
- [17] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, 2008.
- [18] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of ICML*, pages 928–936, 2003.

The Online Loop-free Stochastic Shortest-Path Problem

Gergely Neu^{*†}

neu.gergely@gmail.com

^{*}Department of Computer Science
and Information Theory,
Budapest University of
Technology and Economics

András György[†]

gya@szit.bme.hu

[†]Machine Learning Research
Group, Computer and Automation
Research Institute of the
Hungarian Academy of Sciences

Csaba Szepesvári^{‡†}

szepesva@ualberta.ca

[‡]Department of Computing Science,
University of Alberta

Abstract

We consider a stochastic extension of the loop-free shortest path problem with adversarial rewards. In this episodic Markov decision problem an agent traverses through an acyclic graph with random transitions: at each step of an episode the agent chooses an action, receives some reward, and arrives at a random next state, where the reward and the distribution of the next state depend on the actual state and the chosen action. We consider the bandit situation when only the reward of the just visited state-action pair is revealed to the agent. For this problem we develop algorithms that perform asymptotically as well as the best stationary policy in hindsight. Assuming that all states are reachable with probability $\alpha > 0$ under all policies, we give an algorithm and prove that its regret is $\mathcal{O}(L^2 \sqrt{T} |\mathcal{A}| / \alpha)$, where T is the number of episodes, \mathcal{A} denotes the (finite) set of actions, and L is the length of the longest path in the graph. Variants of the algorithm are given that improve the dependence on the transition probabilities under specific conditions. The results are also extended to variations of the problem, including the case when the agent competes with time varying policies.

1 Introduction

Consider the problem of controlling an inventory so as to maximize the revenue. This is an optimal control problem, where the state of the controlled system is the stock, the action is the amount of stock ordered. The evolution of the stock is also influenced by the demand, which is assumed to be stochastic. Further, the revenue depends on the prices at which products are bought and sold. By assumption, the prices are not available at the time when the decisions are made. Since the prices can depend on many external, often unobserved events, their evolution is often hard to model. Then, a better approach might be to view this problem as an instance of robust control, which can be formulated as follows: Choose a sufficiently large class of controllers so that no matter how the prices evolve, the class contains some controller whose performance is acceptable. The problem is to design an algorithm that is able to perform almost as well as the best controller in the chosen class, where the mentioned best controller is selected based on hindsight.

This problem formulation shares many similarities with the so-called expert framework, where the task is to find an algorithm that can predict (almost) as well as the best amongst a fixed set of experts in an arbitrary prediction environment (cf. Chapter 2 of Cesa-Bianchi and Lugosi, 2006 and the references therein). However, the control problem is made more complicated by the fact that one must take into account that the decisions of the controller influence future states and thus also future rewards. This, in fact, has two consequences: Firstly, in order to perform well, the controller must plan ahead in time. That is, the controller must address the usual temporal credit assignment problem. This is usually done by resorting to some form of (approximate) dynamic programming to maintain computational efficiency (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998). Secondly, the controller must also address the exploration-exploitation problem which arises because only the rewards associated with the state-action pairs visited are available for measurement. This is again made difficult by the fact that in order to be able to explore an action in a given state, the state must first be visited, which requires some planning.

In this paper we consider a special case of this general problem, which we call *the online, loop-free stochastic shortest-path (Online SSP, O-SSP) problem*. This problem is a generalization of two previously considered problems: it is an online extension of the (loop-free version of the) stochastic shortest-path problem (Bertsekas and Tsitsiklis, 1996) and a stochastic extension of the online shortest path problem (György et al., 2007). The problem is defined as follows: The controlled dynamics is stochastic. It is assumed that

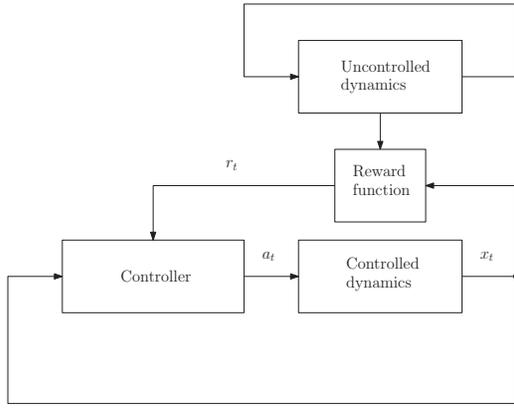


Figure 1: Illustration of the general problem whose special case is studied in the paper: The controlled system has two components. One component, whose state is controlled and observable and is perfectly known, while the other component is unknown and uncontrolled. The second component influences the rewards received and the rewards represent the only source of information about this component. When the uncontrolled part has a complex dynamics and/or a complex state, its identification is hopeless and one might be better off with implementing a robust optimal control strategy, such as the one described in this paper.

the number of states and actions is finite. There is a distinguished initial state and terminal state amongst the states and the state space has the structure of a layered graph: an action chosen at some state of some layer of the graph leads to another state in the next layer. When the terminal state is reached, a new episode starts: the state of the system is reset to the initial state. At the same time, a new reward function is chosen (since no state is visited twice, there is no reason to change the reward function before the end of an episode). Note that only the reward of the last state-action pair is made available to the algorithm, that is, we consider the so-called *bandit setting*. The class of controllers that our algorithm must compete with is selected to be the class of state-feedback policies, that is, policies that select actions according to the actual state, or the class of policies which switch between such state-feedback policies.

Clearly, the inventory management problem mentioned beforehand falls into this class provided that we restrict our attention to its finite horizon variant when the stocks, orders and demand are measured in discrete units, the size of the inventory is limited to lie between a maximum size and zero (excess demands are lost) and where the demands are independent, identically distributed random variables. The O-SSP setup is also particularly suited to address the problem of robust adaptive routing in virtual networks over some (possibly wireless) base network with a fixed routing strategy. Other examples include machine maintenance, asset pricing, or production planning. In general, our framework captures operations research problems, where the control objective involves components which depend on some exogenously developing, hard to model prices.

The main results of this paper are as follows: Assuming that all states are reachable with probability $\alpha > 0$ under all policies, we give an algorithm and prove that its regret is $\mathcal{O}(L^2 \sqrt{T} |\mathcal{A}| / \alpha)$, where L is the number of layers, T is the number of episodes and \mathcal{A} denotes the (finite) set of actions (Theorem 4). Although the number of states in a given layer does not show up in the bound, the bound shows a scaling that is at least linear with the number states since $\max_{1 \leq l \leq L} |\mathcal{X}_l| \leq 1/\alpha$, where $|\mathcal{X}_l|$ is the number of states in the l -th layer. We also give a variant of this result that shows a possibly improved dependence on the transition probabilities (since α can be exponentially small in the size of the number of states). This result is given in Theorem 5. The results are also extended to compete with time-varying policies in Theorem 8. A nice property of the algorithms proposed is that they use bandit algorithms developed for the prediction (stateless) setting, the only requirement for the bandit algorithm being that it should return a probability distribution over the actions. Hence, our algorithm can make use of specially tailored, improved bandit algorithms, for example, algorithms with adaptive tuning that may achieve better performance (and bounds) when the best action has very large gains (Auer et al., 2002b) or algorithms with improved performance when *many* actions have relatively good performance (Exercise 2.6 of Cesa-Bianchi and Lugosi 2006). Specifically, when the **Exp3** algorithm of Auer et al. (2002a) is used, the dependence on α can be improved to $\mathcal{O}(1/\sqrt{\alpha})$ (Theorem 6). Finally, in this special case, under the less stringent assumption that for every state there is some policy that reaches the state with positive probability we give an algorithm whose expected regret per step vanishes over time (Theorem 7).

How do our results compare to those in earlier works in the online learning literature? As noted earlier, our work can be viewed as a stochastic extension of works that considered online shortest path problems in deterministic settings. Here, the closest to our ideas and algorithm is the paper by György et al. (2007). One major difference between the algorithms is that our algorithm is based on direct estimates of the *total* reward to go in every state-action pair, whereas the algorithm of György et al. (2007) estimates the reward to go via estimating the *immediate* rewards. Compared to the bound in György et al. (2007), our bounds are slightly larger (and thus weaker). In earlier work, Awerbuch and Kleinberg (2004) gave an $\mathcal{O}(T^{2/3})$ regret bound, while McMahan and Blum (2004) gave an $\mathcal{O}(T^{3/4})$ bound, building upon the exponentially weighted average forecaster and, respectively, the follow the perturbed leader algorithm, both under the assumption

that the only information received is the total reward at the end of the episodes. More recently, Dani et al. (2008) proposed a generalization of **Exp3** due to Auer et al. (1995), which can be applied to this setting and which gives an expected regret of $\mathcal{O}(|\mathcal{X}|^{3/2}T^{1/2})$, where $|\mathcal{X}|$ is the size of the state space. More recently, Bartlett et al. (2008) showed that the algorithm can be extended so that the bound holds with high probability. We note in passing that Dani et al. (2008) suggest that their algorithm can be implemented efficiently for the MDP setting. However, this is not clear at all: Although, conceptually, the algorithm can be applied to our case, when policies are represented through the distributions that they induce over the state space, but this does not seem to lead to an algorithm that can be implemented.

Another thread of work that is closely related to ours considers algorithms for learning and acting in Markovian decision processes (MDPs) with arbitrary reward sequences. In fact, clearly, our framework is a special case of this more general framework. The first work that considered this setting is due to Even-Dar et al. (2005, 2009). In this work the restriction on the MDP is that it must be *unichain* (i.e., all stationary policies must generate a unique stationary distribution) and it is assumed that the worst mixing time, τ , over all policies is uniformly small (the mixing time appears in the bounds). This is similar to our assumption of the MDP being episodic, with all policies terminating after L steps (though strictly speaking, their assumption does not hold true in our setting). However, the major difference between our work and that of Even-Dar et al. (2005, 2009) is that they assume that the reward function is fully observable, whereas we consider the bandit setting. They propose an algorithm, MDP-E, which is very similar to ours in that it uses some (optimized) expert algorithm in every state which is fed with the action-values of the policy used in the last round (which, in our case, corresponds to the total reward to go). They prove a bound on the expected regret of this algorithm of the form $\mathcal{O}(\tau^2 \sqrt{T \log |\mathcal{A}|})$. The improved dependence on the action set (as compared to our bound stated above) is the result of the assumption that the reward function is available at every step and not only the reward of the last state-action pair visited, otherwise the bound shows a dependence somewhat similar to ours in the main quantities. We actually prove a similar bound for our problem, just to fix some ideas, in Section 4.

More recently, Yu et al. (2009) proposed algorithms for the same (full information) problem and proved a bound on the expected regret of order $\mathcal{O}((\tau + |\mathcal{A}| + |\mathcal{X}|) \tau |\mathcal{A}|^2 T^{3/4+\varepsilon} \log T)$ for arbitrary $\varepsilon \in (0, 1/3)$.¹ The algorithm proposed (“Lazy FPL”) works with phases of length $m^{1/3-\varepsilon}$ and changes policies only at the end of the phases. At the end of a phase the optimal (differential) value function corresponding to the sum of past reward functions is first found. Within the phase, the action to be followed at some time step is then selected as the one that maximizes the one-step lookahead action value computed with this value function but with the immediate rewards perturbed randomly in an appropriate manner. The advantage of this algorithm to that of Even-Dar et al. (2009) is that it is computationally less expensive, which, however, comes at the price of an increased bound on the regret. Yu et al. (2009) introduced another algorithm (“Q-FPL”) which is shown to enjoy a vanishing average regret over time (i.e., the algorithm is Hannan consistent). The major advance, however, is that, for the first time, Yu et al. (2009) proposed an algorithm (“Exploratory FPL”) to address the problem of learning in the bandit setting. This algorithm estimates the immediate rewards by appropriately weighting the rewards received and in a phase either uses a uniformly exploring policy or that of underlying their Lazy FPL algorithm. They prove that the average regret of this algorithm vanishes almost surely.

Yu and Mannor (2009a,b) considered the problem of on-line learning in MDPs where the transition probabilities may also change arbitrarily after each transition. This problem is significantly more difficult than the case where only the reward function is changed arbitrarily. In particular, as it is shown in these papers, Hannan consistency cannot be achieved in this setting. Yu and Mannor (2009b) also considered the case when rewards are only observed along the trajectory traversed by the agent. However, this paper seems to have gaps: If the state space consists of a single state, the problem becomes identical to the non-stochastic multi-armed bandit problem. Yet, from Theorem IV.1 of Yu and Mannor (2009b) it follows that the expected regret of their algorithm is $\mathcal{O}(\sqrt{\log |\mathcal{A}|T})$, which contradicts the known $\Omega(\sqrt{|\mathcal{A}|T})$ lower bound on the regret (Auer et al., 2002a).²

2 Problem definition

Formally, a Markovian Decision Process (MDP) M is defined by a state space \mathcal{X} , an action set \mathcal{A} , a transition function $P : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \rightarrow [0, 1]$, and a reward function $r : \mathcal{X} \times \mathcal{A} \rightarrow [0, 1]$. In time step k , knowing the state $x_k \in \mathcal{X}$, a decision maker (or agent) acting in the MDP M , chooses an action $a_k \in \mathcal{A}(x)$ where $\mathcal{A}(x) \subset \mathcal{A}$ is the set of admissible actions at state x . As a result the process moves to state $x_{k+1} \in \mathcal{X}$ with probability $P(x_{k+1}|x_k, a_k)$ and the decision maker receives reward $r(x_k, a_k)$ (this implies that for any $x \in \mathcal{X}$

¹The notion of mixing time in this paper is somewhat, but not essentially different than that of used by Even-Dar et al. (2005, 2009).

²To show this contradiction note that the condition $T > N$ in the bound of Theorem IV.1 of Yu and Mannor (2009b) can be traded for an extra $\mathcal{O}(1/T)$ term in the regret bound. Then the said contradiction can be arrived at by letting ε, δ converge to zero such that $\varepsilon/\delta^3 \rightarrow 0$.

and $a \in \mathcal{A}(x)$, $P(\cdot|x, a)$ defines a probability distribution over \mathcal{X} . The goal of the agent is to maximize its average reward. In an episodic MDP there is a terminal state $x \in \mathcal{X}$: if this state is reached, the episode is ended and the whole process starts again with a designated starting state. For a more detailed introduction the reader is referred to, for example, Puterman (1994).

The loop-free stochastic shortest path (SSP) problem is a special case of episodic MDPs. Informally, given an acyclic directed graph an agent has to traverse repeatedly over paths between two given vertices of the graph. At each vertex the agent makes a decision, and based on the decision it follows a random edge of the graph to the next vertex and receives some reward. The goal of the agent is to maximize its average reward received over the paths. More formally, we consider MDPs where the state space \mathcal{X} consists of layers, that is, $\mathcal{X} = \cup_{l=0}^L \mathcal{X}_l$, where \mathcal{X}_l is called the l th layer of the state space and $\mathcal{X}_l \cap \mathcal{X}_k = \emptyset$ for all $l \neq k$. The first and last layers are singleton layers, that is, $\mathcal{X}_0 = \{x_0\}$ and $\mathcal{X}_L = \{x_L\}$. The significance of the layers is given by the fact that the state of the agent can only move between consecutive layers, that is, in each episode the agent starts at layer 0, and at time instant l it is at layer l until it reaches the terminal state x_L . This assumption is equivalent to assuming that each path in the graph is of equal length, and is reflected by the special structure of the transition function: for any $x_l \in \mathcal{X}_l$ and $a \in \mathcal{A}(x_l)$, $P(x_{l+1}|x_l, a) = 0$ if $x_{l+1} \notin \mathcal{X}_{l+1}$.³ For any state $x \in \mathcal{X}$ we will use l_x to denote the index of the layer x belongs to, that is, $l_x = l$ if $x \in \mathcal{X}_l$.

In this paper we consider the online version of the loop-free SSP problem, in which case the reward function is allowed to change between episodes, that is, instead of a single reward function r , we are given a sequence of rewards $\{r_t\}$ describing the rewards at episode t that is assumed to be an individual sequence fixed in advance⁴, that is, no statistical assumption is made about the reward values. Note that the constraint that r_t depends only on the current state and action is assumed only for simplicity: the results of the paper can easily be extended to the situation where r_t is allowed to depend on the next state as well (i.e., when the reward function is of the form $r_t(x_l, a_l, x_{l+1})$).

A stochastic stationary policy (or, in short: a policy) is a mapping $\pi : \mathcal{A} \times \mathcal{X} \rightarrow [0, 1]$, where $\pi(a|x) \equiv \pi(a, x)$ is the probability of taking action a in state x . The *instantaneous value function* and *action-value function* with respect to π at episode t are defined, respectively, as

$$v_t^\pi(x_l) = \mathbb{E} \left[\sum_{k=l}^{L-1} r_t(\mathbf{x}_k, \mathbf{a}_k) \mid \mathbf{x}_l = x_l \right]$$

$$q_t^\pi(x_l, a_l) = \mathbb{E} \left[\sum_{k=l_x}^{L-1} r_t(\mathbf{x}_k, \mathbf{a}_k) \mid \mathbf{x}_l = x_l, \mathbf{a}_l = a_l \right],$$

where the sequence $(\mathbf{x}_0, \mathbf{a}_0), (\mathbf{x}_1, \mathbf{a}_1), \dots, (\mathbf{x}_{L-1}, \mathbf{a}_{L-1})$ is generated by the policy π and the MDP, and the expectations are taken with respect to π and the transition function P . These values are equivalently defined by the Bellman equations:

$$q_t^\pi(x, a) = r_t(x, a) + \sum_{x'} P(x'|x, a) v_t^\pi(x')$$

$$v_t^\pi(x) = \sum_a \pi(a|x) q_t^\pi(x, a),$$
(1)

with $v_t^\pi(x_L) = 0$. The *cumulative action-value* and *cumulative value functions* are defined, respectively, as

$$Q_t^\pi = \sum_{s=1}^t q_s^\pi \quad \text{and} \quad V_t^\pi = \sum_{s=1}^t v_s^\pi.$$

Each policy generates a probability distribution μ_π over each layer \mathcal{X}_l , $l = 0, 1, \dots, L$, that is,

$$\mu_\pi(x_l) = \mathbb{P}[\mathbf{x}_l = x_l | \mathbf{x}_0 = x_0].$$

The distribution μ_π can be computed recursively as

$$\mu_\pi(x_l) = \sum_{x_{l-1}, a_{l-1}} P(x_l|x_{l-1}, a_{l-1}) \pi(a_{l-1}|x_{l-1}) \mu_\pi(x_{l-1}),$$
(2)

for $l = 1, 2, \dots, L$, with $\mu_\pi(x_0) = 1$. The *expected return* of a fixed policy π for a time horizon $T > 0$ is defined as $R_T^\pi = \sum_{t=1}^T v_t^\pi = V_T^\pi$. The return of the best policy in hindsight is given by

$$R_T^* = \sup_\pi \sum_{t=1}^T v_t^\pi(x_0) = \sup_\pi V_T^\pi(x_0).$$

³Note that all loop-free state spaces can be transformed to one that satisfies our assumptions. A simple transformation algorithm is given in Appendix A of György et al. (2007).

⁴That is, we assume that we are dealing with a so called oblivious opponent.

It is known that there exists a stationary and deterministic policy π_T^* that achieves the above maximum (Puterman, 1994, Theorem 4.4.2), and so we can use \max instead of \sup in the above equation. By a slight abuse of the notation we will use $\pi_T^*(x)$ to denote the action for which $\pi_T^*(a|x) \neq 0$. The state distribution generated by the optimal policy will be denoted as $\mu_T^* \equiv \mu_{\pi_T^*}$.

Our goal is to construct a sequential decision algorithm (agent) that asymptotically achieves the above return averaged over the episodes. The decision algorithm may follow a different policy π_t at each episode $t = 1, 2, \dots, T$. This policy may be random, as it may depend on the previous states the agent visited and the previous rewards it received. The random path traversed by the agent at episode t will be denoted by

$$\mathbf{u}_t = \left\{ \mathbf{x}_0^{(t)}, \mathbf{a}_0^{(t)}, \mathbf{x}_1^{(t)}, \mathbf{a}_1^{(t)}, \dots, \mathbf{x}_{L-1}^{(t)}, \mathbf{a}_{L-1}^{(t)}, \mathbf{x}_L^{(t)} \right\},$$

and the path history up to episode t by

$$\mathbf{U}_t = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_t\},$$

for all $t = 1, 2, \dots, T$ with $\mathbf{U}_0 = \emptyset$. Note that \mathbf{U}_t covers all the randomness in the problem (including the random transitions and the possible randomness in the agent's decisions). Thus,

$$\pi_t(a|x) = \mathbb{P}[\mathbf{a} = a | \mathbf{x} = x, \mathbf{U}_{t-1}].$$

The value function and the action-value function of policy π_t are given, respectively, by

$$\begin{aligned} \mathbf{v}_t(x_l) &= \mathbb{E} \left[\sum_{k=l_x}^{L-1} r_t(\mathbf{x}_k, \mathbf{a}_k) \middle| \mathbf{x}_l = x_l, \mathbf{U}_{t-1} \right] \\ \mathbf{q}_t(x_l, a_l) &= \mathbb{E} \left[\sum_{k=l_x}^{L-1} r_t(\mathbf{x}_k, \mathbf{a}_k) \middle| \mathbf{x}_l = x_l, \mathbf{a}_l = a_l, \mathbf{U}_{t-1} \right] \end{aligned}$$

where the sequence $(\mathbf{x}_0, \mathbf{a}_0), (\mathbf{x}_1, \mathbf{a}_1), \dots, (\mathbf{x}_{L-1}, \mathbf{a}_{L-1})$ is generated by the policy π_t (that is fully determined by \mathbf{U}_{t-1}). We will also use $\mathbf{Q}_t = \sum_{s=1}^t \mathbf{q}_s$ and $\mathbf{V}_t = \sum_{s=1}^t \mathbf{v}_s$. The state distribution generated by π_t is denoted by $\mu_t = \mu_{\pi_t}$, where $\mu_{\pi_t}(x) = \mathbb{P}[x \in \mathbf{u}_t | \mathbf{U}_{t-1}]$.

The expected return accumulated by the agent in the first T episodes is

$$\hat{R}_T = \sum_{t=1}^T \mathbb{E}[\mathbf{v}_t(x_0)] = \mathbb{E}[\mathbf{V}_T(x_0)],$$

and its relative loss with respect to the best fixed policy π_T^* in hindsight, called *regret*, is defined as

$$\hat{L}_T = R_T^* - \hat{R}_T = V_T^*(x_0) - \mathbb{E}[\mathbf{V}_T(x_0)].$$

The following lemma will be a key to our main results. Note that a similar argument is used by Even-Dar et al. (2009) to prove their main result about online learning in unichain MDPs in the full information case (cf. Lemma 4.1). The benefit of this lemma is that the problem of bounding the regret is essentially reduced to the problem of bounding the difference between action-values of the policy followed by the agent.

Lemma 1 For any time horizon $T > 0$, let the state distribution generated by the optimal policy π_T^* be denoted by μ_T^* , and define

$$V_T^+(x) = \mathbb{E}[\mathbf{Q}_T(x, \pi_T^*(x))].$$

Then

$$V_T^*(x_0) - \mathbb{E}[\mathbf{V}_T(x_0)] = \sum_{l=0}^{L-1} \sum_{x_l \in \mathcal{X}_l} \mu_T^*(x_l) (V_T^+(x_l) - \mathbb{E}[\mathbf{V}_T(x_l)]).$$

Proof:

$$\begin{aligned} V_T^*(x_0) - \mathbb{E}[\mathbf{V}_T(x_0)] &= V_T^*(x_0) - V_T^+(x_0) + V_T^+(x_0) - \mathbb{E}[\mathbf{V}_T(x_0)] \\ &= Q_T^*(x_0, \pi_T^*(x_0)) - \mathbb{E}[Q_T(x_0, \pi_T^*(x_0))] + V_T^+(x_0) - \mathbb{E}[\mathbf{V}_T(x_0)] \\ &= \sum_{x_1 \in \mathcal{X}_1} P(x_1|x_0, \pi_T^*(x_0)) (V_T^*(x_1) - \mathbb{E}[\mathbf{V}_T(x_1)]) + V_T^+(x_0) - \mathbb{E}[\mathbf{V}_T(x_0)] \\ &= \dots = \sum_{l=0}^{L-1} \sum_{x_l \in \mathcal{X}_l} \mu_T^*(x_l) (V_T^+(x_l) - \mathbb{E}[\mathbf{V}_T(x_l)]). \end{aligned}$$

■

3 Sequential prediction with expert advice

A widely studied special case of our setting where the state space consists of a single state is called sequential prediction with expert advice (Cesa-Bianchi and Lugosi, 2006). In this context, actions are usually referred to as *experts*, and several algorithms have been developed that solve the many variants of the problem. Such algorithms E satisfy a regret bound of the form

$$\hat{L}_T \leq \rho_E(T, \mathcal{A}) \quad (3)$$

where $\rho_E(T, \mathcal{A})$ is a sublinear function of T , and so $\lim_{T \rightarrow \infty} \hat{L}_T/T \rightarrow 0$. Furthermore, we assume throughout the paper that $\rho_E(T, \mathcal{A})$ is a nondecreasing function of T and $|\mathcal{A}|$. As usually the regret scales linearly with the range of the rewards, it is assumed above that $r_t \in [0, 1]$. In the course of solving our O-SSP problem we are going to use such algorithms as basic building blocks. Note that depending on the actual form of the algorithm, E may be universal in the sense that (3) is satisfied for all T , while several algorithms require T -dependent parameter settings. On the other hand, these methods can be changed to be universal (sometimes at the price of slightly deteriorating the bounds) with either adaptively changing the parameters or simply by resorting to the doubling trick.

The type of the sequential decision problem is usually classified based on the amount of information available to the decision maker, the set of the reference experts and the way the rewards are generated. In the basic setup, known as the case of the *oblivious opponent*, the reward functions r_1, r_2, \dots are fixed in advance, while in the more general *non-oblivious* setup the rewards may depend on any quantity that is determined before round t . In the latter case, formally we have $\mathbf{r}_t = r_t(\mathbf{U}_{t-1})$.

Luckily, the following lemma, which can be obtained as a special case of a slight generalization of the first part of Lemma 4.1 of Cesa-Bianchi and Lugosi (2006), shows that algorithms that work in the oblivious case also work in the non-oblivious setting:

Lemma 2 *Consider a randomized algorithm E such that, for every $t = 1, 2, \dots, T$, π_t is fully determined by the history \mathbf{U}_{t-1} and the reward sequence $\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_{t-1}$. Assume that the regret of the algorithm satisfies (3) in the oblivious case. Then (3) also holds in the non-oblivious case.*

Note that the regret in the non-oblivious case is still defined as $\max_{a \in \mathcal{A}} \sum_{t=1}^T (\mathbf{r}_t(a) - \mathbf{r}_t(\mathbf{a}_t))$, where $\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_T : \mathcal{A} \rightarrow \mathbb{R}$ are the reward functions that are obtained as a result of following E and \mathbf{a}_t is the action taken by E at time step t . In particular, this definition does not take into account that the sequence of reward functions would be different if action a was followed from the beginning. Although this makes, in general, questionable the meaningfulness of this regret definition, in our case this regret definition will still be just good enough.

In the *full information* case the decision maker is informed about the rewards of all actions at the end of each episode; while in the *bandit setting* only the reward of the chosen action is revealed. An *optimized best expert algorithm* in the full information case is an algorithm that attains an expected regret of $\mathcal{O}(\sqrt{T \log |\mathcal{A}|})$, and similarly, an *optimized $|\mathcal{A}|$ -armed bandit algorithm* is one that attains an expected regret of $\mathcal{O}(\sqrt{T |\mathcal{A}|})$. Optimized best expert algorithms include the *exponentially weighted average forecaster* (EWA) (a variant of Littlestone and Warmuth's (1994) weighted majority algorithm, and Vovk's (1990) aggregating strategies, also known as Hedge (Freund and Schapire, 1997)) and the *follow the perturbed leader* (FPL) algorithm (Kalai and Vempala, 2003). There exist a number of algorithms for the bandit case that attain regrets of $\mathcal{O}(\sqrt{T |\mathcal{A}| \log |\mathcal{A}|})$, such as **Exp3** by Auer et al. (2002a) and **Green** by Allenberg et al. (2006), while the algorithm presented by Audibert and Bubeck (2009) achieves the optimal rate $\mathcal{O}(\sqrt{T |\mathcal{A}|})$.

4 Full information O-SSP

In this section we give an algorithm and a very short proof that bounds the algorithm's regret in the full information case. The purpose is mainly to fix some ideas that will be useful later on.

In the full information case the reward function r_t is completely revealed after each episode t . We will use the value functions of the agent's policy at each episode t to construct the policy in the next round. Note that as we can exactly compute these value functions, the sequence of the agent's policies does not depend on previous decisions, that is, the policies and the value functions are fully determined by the algorithm. Algorithm 1 uses an arbitrary (optimized) best expert algorithm E in each state x to predict the actions to be taken at that state based on previous values of $q_t(x, \cdot)$. (Thus, the algorithm is essentially the same as the MDP-E algorithm of Even-Dar et al. 2009.)

In order to understand how the algorithm works, consider some fixed state x . By definition, $\pi_{t+1}(\cdot|x)$ is the distribution computed by the expert algorithm $E(x)$ when used on a discrete prediction problem with the "reward sequence" $q_1(x, \cdot), q_2(x, \cdot), \dots$ and action set $\mathcal{A}(x)$. Since $q_t(x, \cdot)$ depends on π_t , which depends on the past rewards, the prediction problem is modeled as one with non-oblivious opponents. The cumulative

Algorithm 1 Algorithm for the full information O-SSP.

1. Initialize an expert algorithm $E(x)$, an instance of algorithm E , for all states $x \in \mathcal{X}$.
 2. For $t = 1, 2, \dots, T$, repeat
 - (a) For all $x \in \mathcal{X}$ and all $a \in \mathcal{A}$, let $\pi_t(a|x)$ be the probability that algorithm $E(x)$ chooses action a .
 - (b) Traverse a path \mathbf{u}_t following the policy π_t .
 - (c) Observe the reward function r_t .
 - (d) Compute q_t using the Bellman equations (1) for π_t and r_t .
 - (e) For all states $x \in \mathcal{X}$, feed the algorithm $E(x)$ with $q_t(x, \cdot)$.
-

expected reward of the algorithm up to episode T is $\mathbf{V}_T(x)$ and the reward of a constant action a is $\mathbf{Q}_T(x, a)$. Let E be a best expert algorithm with regret bound $\rho_E(T, \mathcal{A})$. By Lemma 2, for any action a at state x , we get

$$\mathbf{Q}_T(x, a) - \mathbf{V}_T(x) \leq (L - l_x)\rho_E(T, \mathcal{A}),$$

where we used that $0 \leq q_t(x, a) \leq L - l_x$. Since in this case \mathbf{Q}_T is non-random, $V_T^+(x) = \mathbf{Q}_T(x, \pi_T^*(x))$ and thus

$$V_T^+(x) - \mathbf{V}_T(x) \leq (L - l_x)\rho_E(T, \mathcal{A}). \quad (4)$$

Based on this bound and Lemma 1, we immediately obtain a performance bound on this algorithm for our original problem:

Proposition 3 *Let E be an expert algorithm with regret bound $\rho_E(T, \mathcal{A})$. Then the regret of Algorithm 1 can be bounded as*

$$\hat{L}_T \leq \frac{L(L+1)}{2}\rho_E(T, \mathcal{A})$$

Remark: Applying EWA with (time-horizon dependent) optimized parameters as the expert algorithm E , the above bound becomes⁵

$$\hat{L}_T \leq \frac{L(L+1)}{2} \sqrt{\frac{T \log |\mathcal{A}|}{2}}.$$

Proof: By Lemma 1, we have

$$\hat{L}_T = \sum_{l=0}^{L-1} \sum_{x_l \in \mathcal{X}_l} \mu_T^*(x_l) (V_T^+(x_l) - \mathbb{E}[\mathbf{V}_T(x_l)]).$$

Using (4) to bound the terms on the right hand side yields the desired bound. ■

5 Bandit O-SSP

In the bandit case, the rewards are only observed on the paths that the agent traverses at each episode t . In this section we give an algorithm and analyze its performance for this case.

First, we define conditionally unbiased estimates of \mathbf{q}_t and \mathbf{v}_t given \mathbf{U}_{t-1} as follows:

$$\hat{\mathbf{q}}_t(x_l, a_l) = \begin{cases} \frac{\sum_{k=l}^{L-1} r_t(\mathbf{x}_k^{(t)}, \mathbf{a}_k^{(t)})}{\pi_t(a_l|x_l)\mu_t(x_l)} & \text{if } (x_l, a_l) = (\mathbf{x}_l^{(t)}, \mathbf{a}_l^{(t)}); \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

$$\hat{\mathbf{v}}_t(x_l) = \sum_a \pi_t(a|x_l)\hat{\mathbf{q}}_t(x_l, a). \quad (6)$$

Indeed, it is easy to check that $\mathbb{E}[\hat{\mathbf{q}}_t(x, a)|\mathbf{U}_{t-1}] = \mathbf{q}_t(x, a)$ and $\mathbb{E}[\hat{\mathbf{v}}_t(x)|\mathbf{U}_{t-1}] = \mathbf{v}_t(x)$. Note that the estimates $\hat{\mathbf{q}}_t$ and $\hat{\mathbf{v}}_t$ can only be computed after the end of episode t . We will also use the following key property of this estimate:

$$\mathbb{E}[\hat{\mathbf{q}}_t(x, a) - \hat{\mathbf{v}}_t(x)|\mathbb{I}_{x \in \mathbf{u}_t}, \mathbf{U}_{t-1}] = \mathbb{I}_{x \in \mathbf{u}_t} \frac{\mathbf{q}_t(x, a) - \mathbf{v}_t(x)}{\mu_t(x)}. \quad (7)$$

⁵See Theorem 2.2 in Cesa-Bianchi and Lugosi (2006).

Similarly to the full information case, Algorithm 2 given below employs an $|\mathcal{A}(x)|$ -armed bandit algorithm B in each state x to choose actions using the observations from the previous paths that include x . The only assumption that we make about B is that it works with unbiased estimates of the rewards of the form (5), and its regret scales linearly with the range of the rewards. Note that algorithms like **Exp3** can be redefined to receive unbiased estimates of this form instead of the actual rewards. In the following, we use all bandit algorithms with these updates.

Algorithm 2 Algorithm for the bandit O-SSP.

1. Initialize an $|\mathcal{A}(x)|$ -armed bandit algorithm $B(x)$, an instance of B , for all states $x \in \mathcal{X}$.
 2. For $t = 1, 2, \dots, T$, repeat
 - (a) For all $x \in \mathcal{X}$ and all $a \in \mathcal{A}$, let $\pi_t(a|x)$ be the probability that algorithm $B(x)$ chooses action a .
 - (b) Compute $\mu_t(x)$ for all $x \in \mathcal{X}$ using (2) recursively.
 - (c) Traverse a path \mathbf{u}_t following the policy π_t .
 - (d) Observe rewards $r_t(\mathbf{u}_t) = \left\{ r_t(\mathbf{x}_0^{(t)}, \mathbf{a}_0^{(t)}), \dots, r_t(\mathbf{x}_{L-1}^{(t)}, \mathbf{a}_{L-1}^{(t)}) \right\}$.
 - (e) Construct estimates $\hat{\mathbf{q}}_t$ using equation (5).
 - (f) For all states $x \in \mathcal{X}$, feed the algorithm $B(x)$ with $\hat{\mathbf{q}}_t(x, \cdot)$
-

Theorem 4 *Let B be an multi-armed bandit algorithm with regret bound $\rho_B(T, \mathcal{A})$. Assume that there exists some $\alpha > 0$ for which $\mu_\pi(x) \geq \alpha$ holds for all $x \in \mathcal{X}$ and all stationary policies π . Then the regret of Algorithm 2 can be bounded as*

$$\hat{L}_T \leq \frac{L(L+1)}{2\alpha} \rho_B(T, \mathcal{A}).$$

Remark: For example, using the algorithm of Audibert and Bubeck (2009) with appropriate parameters as the base bandit algorithm B yields

$$\hat{L}_T \leq \frac{15L(L+1)}{2\alpha} \sqrt{T|\mathcal{A}|}.$$

Also note that the conditions of the proposition are satisfied if, for example,

$$\min_{x \in \mathcal{X}_l, a \in \mathcal{A}, x' \in \mathcal{X}_{l+1}, l \in 1:L-1} P(x'|x, a) > 0.$$

In fact, our assumption of α being positive is closely related to the uniform mixing assumption used generally in the literature considering online learning in MDPs.

Proof: The set of episodes when state x is visited will be denoted by $\mathbf{T}_x = \{1 \leq t \leq T \mid x \in u_t\}$. By Lemma 1, we have

$$\hat{L}_T = \sum_{l=0}^{L-1} \sum_{x_l \in \mathcal{X}_l} \mu_T^*(x_l) [V_T^+(x_l) - \mathbb{E}[\mathbf{V}_T(x_l)]] . \quad (8)$$

On the other hand, we have, for any fixed x ,

$$\begin{aligned} V_T^+(x) - \mathbb{E}[\mathbf{V}_T(x)] &= \mathbb{E} \left[\sum_{t=1}^T \mathbb{E} [\hat{\mathbf{q}}_t(x, \pi_T^*(x)) - \hat{\mathbf{v}}_t(x) \mid \mathbf{U}_{t-1}] \right] \\ &= \sum_{t=1}^T \mathbb{E} [\hat{\mathbf{q}}_t(x, \pi_T^*(x)) - \hat{\mathbf{v}}_t(x)] . \end{aligned} \quad (9)$$

Therefore, by (7) we obtain

$$\begin{aligned}
V_T^+(x) - \mathbb{E}[\mathbf{V}_T(x)] &= \mathbb{E} \left[\sum_{t=1}^T \hat{\mathbf{q}}_t(x, \pi_T^*(x)) - \hat{\mathbf{v}}_t(x) \right] \\
&= \mathbb{E} \left[\sum_{t=1}^T \mathbb{E} \left[\hat{\mathbf{q}}_t(x, \pi_T^*(x)) - \hat{\mathbf{v}}_t(x) \mid \mathbb{I}_{x \in \mathbf{u}_t}, \mathbf{U}_{t-1} \right] \right] \\
&= \mathbb{E} \left[\sum_{t=1}^T \mathbb{I}_{x \in \mathbf{u}_t} \frac{\mathbf{q}_t(x, \pi_T^*(x)) - \mathbf{v}_t(x)}{\boldsymbol{\mu}_t(x)} \right] \\
&= \mathbb{E} \left[\sum_{t \in \mathbf{T}_x} \frac{\mathbf{q}_t(x, \pi_T^*(x)) - \mathbf{v}_t(x)}{\boldsymbol{\mu}_t(x)} \right]. \tag{10}
\end{aligned}$$

As for every x we are using an independent $|\mathcal{A}(x)|$ -armed bandit algorithm B with regret bound $\rho_B(T, \mathcal{A}(x))$ that is fed with values $\hat{\mathbf{q}}_t(x, \cdot)$ which are conditionally unbiased estimates of values that belong to $[0, (L - l_x)/\alpha]$, by Lemma 2 we have the following for any fixed a :

$$\mathbb{E} \left[\sum_{t \in \mathbf{T}_x} \frac{\mathbf{q}_t(x, a) - \mathbf{v}_t(x)}{\boldsymbol{\mu}_t(x)} \right] \leq \frac{1}{\alpha} (L - l_x) \rho_B(T, \mathcal{A}(x)) \leq \frac{1}{\alpha} (L - l_x) \rho_B(T, \mathcal{A}).$$

Combining this bound with (8)-(10) finishes the proof. \blacksquare

A problem with the above theorem is that the bound scales with $1/\alpha$, but in certain cases α can be exponentially small. On the other hand, if the minimal probability of visiting a state is exponentially small then the maximal probability of visiting the same state may often be also exponentially small (clearly this is the case in the grid-world example considered in the simulations in Section 6, see Figure 2). The following theorem can be very useful in these situations.

Theorem 5 *Let B be a multi-armed bandit algorithm with regret bound $\rho_B(T, \mathcal{A})$, and define*

$$\alpha(x) = \min_{\pi} \mu_{\pi}(x) \quad \text{and} \quad \beta(x) = \max_{\pi} \mu_{\pi}(x).$$

Assume that $\kappa = \max_{x \in \mathcal{X}} \frac{\beta(x)}{\alpha(x)} < \infty$. Then the regret of Algorithm 2 can be bounded as

$$\hat{L}_T \leq \kappa L |\mathcal{X}| \rho_B(T, \mathcal{A}).$$

Proof: Following the proof of Theorem 4 we obtain, for any l ,

$$\begin{aligned}
\sum_{x_l \in \mathcal{X}_l} \mu_T^*(x_l) (V_T^+(x_l) - \mathbb{E}[\mathbf{V}_T(x_l)]) &= \sum_{x_l \in \mathcal{X}_l} \mu_T^*(x_l) \mathbb{E} \left[\sum_{t \in \mathbf{T}_{x_l}} \frac{\mathbf{q}_t(x_l, \pi_T^*(x_l)) - \mathbf{v}_t(x_l)}{\boldsymbol{\mu}_t(x_l)} \right] \\
&\leq \sum_{x_l \in \mathcal{X}_l} \beta(x_l) \frac{1}{\alpha(x_l)} (L - l) \rho_B(T, \mathcal{A}) \\
&\leq |\mathcal{X}_l| \kappa L \rho_B(T, \mathcal{A}).
\end{aligned}$$

Summing up for all l finishes the proof. \blacksquare

In particular, if we use **Exp3** (as described in Section 6.8 of Cesa-Bianchi and Lugosi 2006) as the bandit algorithm B , we can prove regret bounds that have slightly better dependence on α . The proof of the results, given in the following theorem, follows closely the derivation of the original regret bound of the **Exp3** algorithm (Auer et al., 2002a) and will be given in details in an extended version of this paper.

Theorem 6 *Assume that the conditions of Theorem 4 hold and the bandit algorithm B is the **Exp3** algorithm with parameters $0 < \gamma \leq 1$ and $0 < \eta \leq \frac{\alpha \gamma}{|\mathcal{A}|(L - l_x)}$. Then, if Algorithm 2 is used, for each state $x \in \mathcal{X}$ we have*

$$\mathbb{E}[\mathbf{Q}_T(x, a) - \mathbf{V}_T(x)] \leq \left(\gamma + (e - 2) \eta \frac{L - l_x}{\alpha} |\mathcal{A}| \right) (L - l_x) T + \frac{\ln |\mathcal{A}|}{\eta}.$$

An optimal choice of γ and η yields the following bound on the regret:

$$\hat{L}_T \leq \frac{L(L + 1)}{2} \sqrt{\frac{T |\mathcal{A}| \ln |\mathcal{A}| (e - 2)}{\alpha}}.$$

Furthermore, let $\kappa' = \max_{x \in \mathcal{X}} \frac{\beta(x)}{\sqrt{\alpha(x)}} < \infty$ where $\alpha(x)$ and $\beta(x)$ are defined in Theorem 5. Then

$$\hat{L}_T \leq \kappa' L |\mathcal{X}| \sqrt{T |\mathcal{A}| \ln |\mathcal{A}| (e-2)}.$$

In the above results we used the assumption that any stationary policy induces a distribution that visits each state with positive probability. However, this assumption may be too restrictive in many situations. If we only require that each state is reachable with positive probability for an adequately chosen policy, then using **Exp3** in our algorithm with different γ at each layer yields a consistent strategy with sublinear regret, although the convergence rate becomes very slow.

Theorem 7 *Let*

$$p_{\min} = \min_{x \in \mathcal{X}_l, a \in \mathcal{A}, x' \in \mathcal{X}_{l+1}, 1 \leq l \leq L-1, P(x'|x,a) > 0} P(x'|x, a)$$

and assume that for each state x there is a policy π such that $\mu_\pi(x) > 0$. If Algorithm 2 is run with the **Exp3** algorithm with parameters $\gamma_l = T^{-2^{-l-1}}$ and $\eta_l = \frac{\gamma_l \prod_{i=1}^{l-1} (p_{\min} \gamma_i / |\mathcal{A}|)}{|\mathcal{A}|(L-l)}$ for each state $x_l \in \mathcal{X}_l$, then

$$\hat{L}_T / T \leq \frac{L(L+1)}{2} \left((e-1) + \frac{|\mathcal{A}|^{L+1} \ln |\mathcal{A}|}{p_{\min}^L} \right) T^{-2^{-L-1}}$$

Proof: For any l our assumptions imply $\mu_t(x_l) \geq \prod_{i=0}^{l-1} (p_{\min} \gamma_i / |\mathcal{A}|)$. Therefore, similarly to the first statement of Theorem 6, for all x and a we have

$$\begin{aligned} \mathbb{E}[\mathbf{Q}_T(x, a) - \mathbf{V}_T(x)] &\leq \left(\eta_x + \frac{(e-2)\eta_x(L-l_x)|\mathcal{A}|}{\prod_{i=0}^{l_x-1} (p_{\min} \gamma_i / |\mathcal{A}|)} \right) (L-l_x)T + \frac{\ln |\mathcal{A}|}{\eta_x} \\ &= \left((L-l_x)(e-1) + \frac{(L-l_x)|\mathcal{A}|^{l_x+1} \ln |\mathcal{A}|}{p_{\min}^{l_x}} \right) T^{1-2^{-l_x-1}} \end{aligned}$$

by straightforward calculations. Summing up the above formula for $l_x = 0, \dots, L-1$ proves the proposition by Lemma 1. \blacksquare

So far the regret of our algorithm was measured relative to the best fixed policy. On the other hand, in our motivating examples it may be the case that the best policy changes over time, and hence it is natural to compare our performance to the best time varying policy. Let $\pi_{1:T} = (\pi_1, \pi_2, \dots, \pi_T)$ be a sequence of policies, and let $R_T(\pi_{1:T})$ denote the expected return, after T episodes, of the algorithm that applies policy π_t at episode t . Our goal is to minimize the expected loss $R_T(\pi_{1:T}) - \hat{R}_T$ relative to $\pi_{1:T}$.

Clearly, it is not possible to provide a uniform bound on this loss, as, in general, it is harder to achieve the performance of an algorithm that changes the employed policy more often (the extreme situation is when the policy changes in each time instant). In the following we will give an algorithm that bounds the tracking regret with the help of the complexity of $\pi_{1:T}$ that can be defined as

$$C(\pi_{1:T}) = 1 + |\{t : \pi_t \neq \pi_{t+1}, 1 \leq t \leq T-1\}|.$$

That is $C(\pi_{1:T})$ is the number of times the employed policy changes between consecutive episodes.

While this problem seems much harder than the ones considered before, the tracking algorithms for the prediction framework help us in solving it. Several algorithms are known for the full information case with vanishing tracking regret under various conditions and with different rewards, see, for example, Willems (1996); Helmbold and Warmuth (1998); Shamir and Merhav (1999); Vovk (1999); György et al. (2008). These methods can be extended to the bandit case as well, see, for example, Auer et al. (2002a). Assume that we have an algorithm BT for the bandit sequential prediction problem (that is, when there is only one state) that satisfies, for every policy sequence $\pi_{1:T}$,

$$R_T(\pi_{1:T}) - \hat{R}_T \leq \rho_{BT}(T, \mathcal{A}, C(\pi_{1:T})) \quad (11)$$

with some function $\rho_{BT}(T, \mathcal{A}, C(\pi_{1:T}))$ that is a nondecreasing function of T , \mathcal{A} , and $C(\pi_{1:T})$. Then using such an algorithm as the expert algorithm B in Algorithm 2 solves the tracking problem in the following sense.

Theorem 8 *Assume that BT is a multi-armed bandit algorithm that satisfies the regret bound (11). If κ , defined in Theorem 5, is finite and Algorithm 2 is used with the bandit algorithm BT , then the regret relative to any fixed sequence of policies $\pi_{1:T}$ can be bounded as*

$$R_T(\pi_{1:T}) - \hat{R}_T \leq \kappa L |\mathcal{X}| \rho_{BT}(T, \mathcal{A}, C(\pi_{1:T})).$$

Remark: In particular, if the **Exp3.S** algorithm of Auer et al. (2002a) is used, then if T is known in advance and is used optimally in setting the parameters of the algorithm, we obtain

$$R_T(\pi_{1:T}) - \hat{R}_T \leq \kappa L |\mathcal{X}| \left(C(\pi_{1:T}) \sqrt{|\mathcal{A}|T \ln(|\mathcal{A}|T)} + 2e \sqrt{\frac{|\mathcal{A}|T}{\ln(|\mathcal{A}|T)}} \right).$$

Furthermore, if a bound C on the complexity of $\pi_{1:T}$ is known in advance (this is useful, if the complexity of the optimal $\pi_{1:T}$ is bounded), then using this value in setting the parameters of **Exp3.S**, we obtain

$$R_T(\pi_{1:T}) - \hat{R}_T \leq \kappa L |\mathcal{X}| \sqrt{e-1} \sqrt{|\mathcal{A}|T(C \ln(|\mathcal{A}|T) + e)}.$$

Proof: A simple generalization of Lemma 1 yields

$$\begin{aligned} R_T(\pi_{1:T}) - \hat{R}_T &= V_T^{\pi_{1:T}}(x_0) - \mathbb{E}[\mathbf{V}_T(x_0)] = \sum_{t=1}^T v_t^{\pi_t}(x_0) - \mathbb{E}[\mathbf{V}_T(x_0)] \\ &= \sum_{l=0}^{L-1} \sum_{x_l \in \mathcal{X}_l} \sum_{t=1}^T \mu_t(x_l) \mathbb{E}[\mathbf{q}_t(x_l, \pi_t(x_l)) - \mathbf{v}_t(x_l)] \end{aligned}$$

Now we have, for any x , similarly to (9),

$$\mathbb{E}[\mathbf{q}_t(x, \pi_t(x)) - \mathbf{v}_t(x)] = \mathbb{E}[\hat{\mathbf{q}}_t(x, \pi_t(x)) - \hat{\mathbf{v}}_t(x)].$$

Therefore, similarly to (10), we obtain

$$\begin{aligned} \sum_{t=1}^T \mu_t(x) \mathbb{E}[\mathbf{q}_t(x, \pi_t(x)) - \mathbf{v}_t(x)] &= \sum_{t=1}^T \mu_t(x) \mathbb{E} \left[\frac{\mathbf{q}_t(x, \pi_t(x)) - \mathbf{v}_t(x)}{\boldsymbol{\mu}_t(x)} \right] \\ &\leq \beta(x) \mathbb{E} \left[\sum_{t \in \mathbf{T}_x} \frac{\mathbf{q}_t(x, \pi_t(x)) - \mathbf{v}_t(x)}{\boldsymbol{\mu}_t(x)} \right] \end{aligned}$$

Finally, (11) and Lemma 2 yields, as at the end of the proof of Theorem 4,

$$\mathbb{E} \left[\sum_{t \in \mathbf{T}_x} \frac{\mathbf{q}_t(x, \pi_t(x)) - \mathbf{v}_t(x)}{\boldsymbol{\mu}_t(x)} \right] \leq \frac{L}{\alpha(x)} \rho_{BT}(T, \mathcal{A}, C(\pi_{1:T}))$$

since $\rho_{BT}(T, \mathcal{A}, C)$ is an increasing function of C by assumption. Combining the above results finishes the proof. \blacksquare

6 Simulations

We have run our experiments on a grid world of size 10×10 , where in each episode the agent has to find the shortest path from the lower left corner to the upper right corner. The agent has two actions: Both make the agent move right or up, the “right” (“up”) action makes the agent move right (respectively, “up”) with probability 0.7, while it makes it move “up” (respectively, “right”) with probability 0.3. That is, we have $L = 20$, $|\mathcal{X}| = 100$, $\alpha = 0.3^{10}$, $\kappa = (0.7/0.3)^{10}$ (the values of α and κ correspond to the top-left and bottom-right corners). The experiment is run with $T = 100,000$, rewards are set randomly 20 times at episodes $t = 1, 5000, 10000, \dots$ for all x, a , and change linearly in between. We have simulated the policies generated by EWA for the full information case, and the policies generated by **Exp3** for the bandit case. An example of the grid-world (of smaller size) and the results of a typical simulation are shown in Figure 2.

7 Conclusions and future work

In this paper we considered the problem of online learning in loop-free stochastic-shortest path problems in a bandit setting when only the reward of the current transitions is available for measurement. The per episode complexity of our algorithm is $\mathcal{O}(|\mathcal{A}| |\mathcal{X}|^2)$ and the algorithm is easy to implement. According to our knowledge, ours is the first algorithm that can be implemented efficiently and which is known to achieve an $\mathcal{O}(\sqrt{T|\mathcal{A}|})$ regret in the bandit setting, under the assumption that every policy reaches every state with positive probability. Unfortunately, the regret bound scales with the inverse of the minimal such probability, which is clearly undesirable in many situations. To alleviate this problem, variants of the original bound have been developed that may be preferred in certain specific cases. For the case when this latter condition does not hold, we proposed an algorithm whose expected average expected regret vanishes over time. We view our

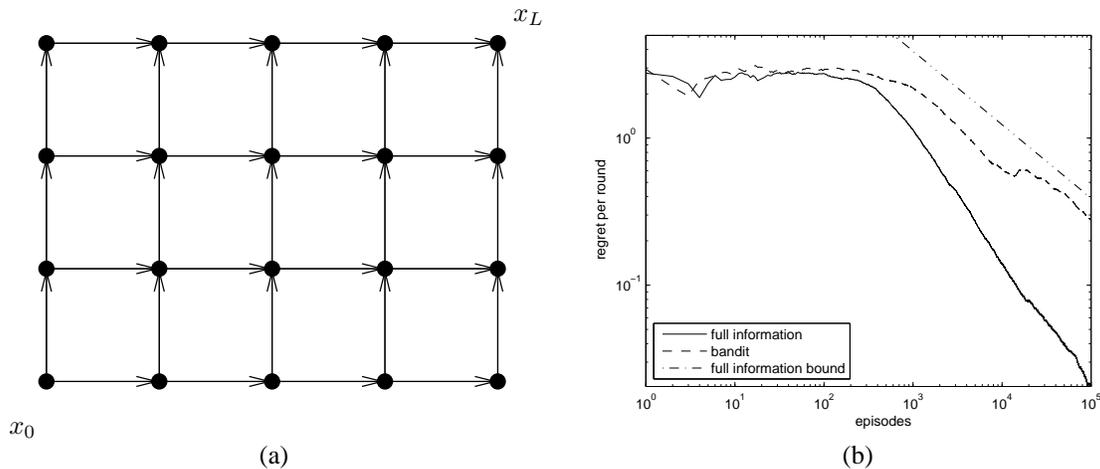


Figure 2: (a) An example of a grid-world. (b) The average regret in an episode of the proposed algorithms as the function of the number of episodes in a simple MDP.

results as a step towards algorithms that work efficiently and which can be implemented efficiently. However, much work remains to be done.

As for immediate future work, obvious directions include extending our results to the case of unichain MDPs setting, or, less ambitiously, to the case when the stochastic shortest-path problem may have loops. Although one can construct an unbiased estimate of the action values by plugging in an unbiased estimate of the rewards, these estimates are not of the form (5), thus our analysis does not apply. It is nontrivial whether a proper estimate of the action values can be found; even with a positive answer there are further obstacles to eliminate (e.g., the change rate of the distributions generated by the applied bandit algorithm has to be controlled in order to be able to apply the analysis of Even-Dar et al., 2009). Alternate directions to extend our results include the case of unknown transition probabilities, partial monitoring, high probability bounds, or when the state and action space are too large to keep a value for each of them, in which case one must resort to some form of function approximation, just to mention a few.

Acknowledgments

This work was supported in part by the PASCAL2 Network of Excellence under EC grant no. 216886, the Hungarian Scientific Research Fund and the Hungarian National Office for Research and Technology (OTKA-NKTH CNK 77782), NSERC, the Alberta Ingenuity Centre for Machine Learning and iCore.

References

- Allenberg, C., Auer, P., Györfi, L., and Ottucsák, G. (2006). Hannan consistency in on-line learning in case of unbounded losses under partial monitoring. In *ALT*, pages 229–243.
- Audibert, J.-Y. and Bubeck, S. (2009). Minimax policies for bandits games. In *COLT 2009*.
- Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. (1995). Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on the Foundations of Computer Science*, pages 322–331. IEEE press.
- Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002a). The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77.
- Auer, P., Cesa-Bianchi, N., and Gentile, C. (2002b). Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 64(1).
- Awerbuch, B. and Kleinberg, R. D. (2004). Adaptive routing with end-to-end feedback: distributed learning and geometric approaches. In Babai, L., editor, *STOC*, pages 45–53. ACM.
- Bartlett, P. L., Dani, V., Hayes, T. P., Kakade, S., Rakhlin, A., and Tewari, A. (2008). High-probability regret bounds for bandit online linear optimization. In Servedio, R. A. and Zhang, T., editors, *21st Annual Conference on Learning Theory (COLT 2008)*, pages 335–342. Omnipress.

- Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.
- Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA.
- Dani, V., Hayes, T. P., and Kakade, S. (2008). The price of bandit information for online optimization. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20*, pages 345–352. MIT Press.
- Even-Dar, E., Kakade, S. M., and Mansour, Y. (2005). Experts in a Markov decision process. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 401–408.
- Even-Dar, E., Kakade, S. M., and Mansour, Y. (2009). Online Markov decision processes. *Mathematics of Operations Research*, 34(3):726–736.
- Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139.
- György, A., Linder, T., and Lugosi, G. (2008). Tracking the best quantizer. *IEEE Transactions on Information Theory*, 54:1604–1625.
- György, A., Linder, T., Lugosi, G., and Ottucsák, G. (2007). The on-line shortest path problem under partial monitoring. *J. Mach. Learn. Res.*, 8:2369–2403.
- Helmbold, D. and Warmuth, M. (1998). Tracking the best expert. *Machine Learning*, 32(2):151–178.
- Kalai, A. and Vempala, S. (2003). Efficient algorithms for the online decision problem. In Schölkopf, B. and Warmuth, M., editors, *Proceedings of the 16th Annual Conference on Learning Theory and the 7th Kernel Workshop, COLT-Kernel 2003*, pages 26–40, New York, USA. Springer.
- Littlestone, N. and Warmuth, M. (1994). The weighted majority algorithm. *Information and Computation*, 108:212–261.
- McMahan, H. B. and Blum, A. (2004). Online geometric optimization in the bandit setting against an adaptive adversary. In Shawe-Taylor, J. and Singer, Y., editors, *COLT*, volume 3120 of *Lecture Notes in Computer Science*, pages 109–123. Springer.
- Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience.
- Shamir, G. I. and Merhav, N. (1999). Low-complexity sequential lossless coding for piecewise-stationary memoryless sources. *IEEE Trans. Inform. Theory*, IT-45:1498–1519.
- Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIP Press.
- Vovk, V. (1990). Aggregating strategies. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pages 372–383.
- Vovk, V. (1999). Derandomizing stochastic prediction strategies. *Machine Learning*, 35(3):247–282.
- Willems, F. M. J. (1996). Coding for a binary independent piecewise-identically-distributed source. *IEEE Trans. Inform. Theory*, IT-42:2210–2217.
- Yu, J. Y. and Mannor, S. (2009a). Arbitrarily modulated Markov decision processes. In *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*. IEEE Press.
- Yu, J. Y. and Mannor, S. (2009b). Online learning in Markov decision processes with arbitrarily changing rewards and transitions. In *GameNets'09: Proceedings of the First ICST international conference on Game Theory for Networks*, pages 314–322, Piscataway, NJ, USA. IEEE Press.
- Yu, J. Y., Mannor, S., and Shimkin, N. (2009). Markov decision processes with arbitrary reward processes. *Mathematics of Operations Research*, 34(3):737–757.

Adaptive Bound Optimization for Online Convex Optimization

H. Brendan McMahan
Google, Inc.
mcmahan@google.com

Matthew Streeter
Google, Inc.
mstreeter@google.com

Abstract

We introduce a new online convex optimization algorithm that adaptively chooses its regularization function based on the loss functions observed so far. This is in contrast to previous algorithms that use a fixed regularization function such as L_2 -squared, and modify it only via a single time-dependent parameter. Our algorithm's regret bounds are worst-case optimal, and for certain realistic classes of loss functions they are much better than existing bounds. These bounds are problem-dependent, which means they can exploit the structure of the actual problem instance. Critically, however, our algorithm does not need to know this structure in advance. Rather, we prove competitive guarantees that show the algorithm provides a bound within a constant factor of the best possible bound (of a certain functional form) in hindsight.

1 Introduction

We consider online convex optimization in the full information feedback setting. A closed, bounded convex feasible set $\mathcal{F} \subseteq \mathbb{R}^n$ is given as input, and on each round $t = 1, \dots, T$, we must pick a point $x_t \in \mathcal{F}$. A convex loss function f_t is then revealed, and we incur loss $f_t(x_t)$. Our regret at the end of T rounds is

$$\text{Regret} \equiv \sum_{t=1}^T f_t(x_t) - \min_{x \in \mathcal{F}} \sum_{t=1}^T f_t(x). \quad (1)$$

Existing algorithms for online convex optimization are worst-case optimal in terms of certain fundamental quantities. In particular, online gradient descent attains a bound of $\mathcal{O}(DM\sqrt{T})$ where D is the L_2 diameter of the feasible set and M is a bound on L_2 -norm of the gradients of the loss functions. This bound is tight in the worst case, in that it is possible to construct problems where this much regret is inevitable. However, this does not mean that an algorithm that achieves this bound is optimal in a practical sense, as on easy problem instances such an algorithm is still allowed to incur the worst-case regret. In particular, although this bound is minimax optimal when the feasible set is a hypersphere (Abernethy et al., 2008), we will see that much better algorithms exist when the feasible set is the hypercube.

To improve over the existing worst-case guarantees, we introduce additional parameters that capture more of the problem's structure. These parameters depend on the loss functions, which are not known in advance. To address this, we first construct functional upper bounds on regret $B_R(\theta_1, \dots, \theta_T; f_1, \dots, f_T)$ that depend on both (properties of) the loss functions f_t and algorithm parameters θ_t . We then give algorithms for choosing the parameters θ_t adaptively (based only on f_1, f_2, \dots, f_{t-1}) and prove that these adaptive schemes provide a regret bound that is only a constant factor worse than the best possible regret bound of the form B_R . Formally, if for all possible function sequences f_1, \dots, f_T we have

$$B_R(\theta_1, \dots, \theta_T; f_1, \dots, f_T) \leq \kappa \inf_{\theta'_1, \dots, \theta'_T \in \Theta^T} B_R(\theta'_1, \dots, \theta'_T; f_1, \dots, f_T)$$

for the adaptively-selected θ_t , we say the adaptive scheme is κ -competitive for the bound optimization problem. In Section 1.2, we provide realistic examples where known bounds are much worse than the problem-dependent bounds obtained by our algorithm.

1.1 Follow the proximally-regularized leader

We analyze a *follow the regularized leader* (FTRL) algorithm that adaptively selects regularization functions of the form

$$r_t(x) = \frac{1}{2} \|(Q_t^{\frac{1}{2}}(x - x_t))\|_2^2$$

where Q_t is a positive semidefinite matrix. Our algorithm plays $x_1 = 0$ on round 1 (we assume without loss of generality that $0 \in \mathcal{F}$), and on round $t + 1$, selects the point

$$x_{t+1} = \arg \min_{x \in \mathcal{F}} \left(\sum_{\tau=1}^t (r_\tau(x) + f_\tau(x)) \right). \quad (2)$$

In contrast to other FTRL algorithms, such as the dual averaging method of Xiao (2009), we center the additional regularization at the current feasible point x_t rather than at the origin. Accordingly, we call this algorithm *follow the proximally-regularized leader* (FTPRL). This proximal centering of additional regularization is similar in spirit to the optimization solved by online gradient descent (and more generally, online mirror descent, (Cesa-Bianchi & Lugosi, 2006)). However, rather than considering only the current gradient, our algorithm considers the sum of all previous gradients, and so solves a global rather than local optimization on each round. We discuss related work in more detail in Section 4.

The FTPRL algorithm allows a clean analysis from first principles, which we present in Section 2. The proof techniques are rather different from those used for online gradient descent algorithms, and will likely be of independent interest.

We write \vec{Q}_T as shorthand for (Q_1, Q_2, \dots, Q_T) , with \vec{g}_T defined analogously. For a convex set \mathcal{F} , we define $\mathcal{F}_{\text{sym}} = \{x - x' \mid x, x' \in \mathcal{F}\}$. Using this notation, we can state our regret bound as

$$\text{Regret} \leq B_R(\vec{Q}_T, \vec{g}_T) \equiv \frac{1}{2} \sum_{t=1}^T \max_{\hat{y} \in \mathcal{F}_{\text{sym}}} (\hat{y}^\top Q_t \hat{y}) + \sum_{t=1}^T g_t^\top Q_{1:t}^{-1} g_t \quad (3)$$

where g_t is a subgradient of f_t at x_t and $Q_{1:t} = \sum_{\tau=1}^t Q_\tau$. We prove competitive ratios with respect to this B_R for several adaptive schemes for selecting the Q_t matrices. In particular, when the FTPRL-Diag scheme is run on a hyperrectangle (a set of the form $\{x \mid x_i \in [a_i, b_i]\} \subseteq \mathbb{R}^n$), we achieve

$$\text{Regret} \leq \sqrt{2} \inf_{\vec{Q} \in \mathcal{Q}_{\text{diag}}^T} B_R(\vec{Q}_T, \vec{g}_T)$$

where $\mathcal{Q}_{\text{diag}} = \{\text{diag}(\lambda_1, \dots, \lambda_n) \mid \lambda_i \geq 0\}$. When the FTPRL-Scale scheme is run on a feasible set of the form $\mathcal{F} = \{x \mid \|Ax\|_2 \leq 1\}$ for $A \in S_{++}^n$, it is competitive with arbitrary positive semidefinite matrices:

$$\text{Regret} \leq \sqrt{2} \inf_{\vec{Q} \in (S_{++}^n)^T} B_R(\vec{Q}_T, \vec{g}_T).$$

Our analysis of FTPRL reveals a fundamental connection between the shape of the feasible set and the importance of choosing the regularization matrices adaptively. When the feasible set is a hyperrectangle, FTPRL-Diag has stronger bounds than known algorithms, except for degenerate cases where the bounds are identical. In contrast, when the feasible set is a hypersphere, $\{x \mid \|x\|_2 \leq 1\}$, the bound B_R is always optimized by choosing $Q_t = \lambda_t I$ for suitable $\lambda_t \in \mathbb{R}$. The FTPRL-Scale scheme extends this result to hyperellipsoids by applying a suitable transformation. These results are presented in detail in Section 3.

1.2 The practical importance of adaptive regularization

In the past few years, online algorithms have emerged as state-of-the-art techniques for solving large-scale machine learning problems (Bottou & Bousquet, 2008; Zhang, 2004). Two canonical examples of such large-scale learning problems are text classification on large datasets and predicting click-through rates for ads on a search engine. For such problems, extremely large feature sets may be considered, but many features only occur rarely, while few occur very often. Our diagonal-adaptation algorithm offers improved bounds for problems such as these.

As an example, suppose $\mathcal{F} = [-\frac{1}{2}, \frac{1}{2}]^n$ (so $D = \sqrt{n}$). On each round t , the i th component of $\nabla f_t(x_t)$ (henceforth $g_{t,i}$) is 1 with probability $i^{-\alpha}$, and is 0 otherwise, for some $\alpha \in [1, 2)$. Such heavy-tailed distributions are common in text classification applications, where there is a feature for each word. In this case, gradient descent with a global learning rate¹ obtains an expected regret bound of $O(\sqrt{nT})$. In contrast,

¹The $O(DM\sqrt{T})$ bound (mentioned in the introduction) based on a $1/\sqrt{t}$ learning rate gives $O(n\sqrt{T})$ here; to get $O(\sqrt{nT})$ a global rate based on $\|g_t^2\|$ is needed, e.g., Corollary 8.

the algorithms presented in this paper will obtain expected regret on the order of

$$\mathbb{E} \left[\sum_{i=1}^n \sqrt{\sum_{t=1}^T g_{t,i}^2} \right] \leq \sum_{i=1}^n \sqrt{\sum_{t=1}^T \mathbb{E} [g_{t,i}^2]} = \sum_{i=1}^n \sqrt{T i^{-\alpha}} = O(\sqrt{T} \cdot n^{1-\frac{\alpha}{2}})$$

using Jensen’s inequality. This bound is never worse than the $O(\sqrt{nT})$ bound achieved by ordinary gradient descent, and can be substantially better. For example, in problems where a constant fraction of examples contain a new feature, n is $\Omega(T)$ and the bound for ordinary gradient descent is vacuous. In contrast, the bound for our algorithm is $O(T^{\frac{3-\alpha}{2}})$, which is sublinear for $\alpha > 1$.

This performance difference is not merely a weakness in the regret bounds for ordinary gradient descent, but is a difference in actual regret. In concurrent work (Streeter & McMahan, 2010), we showed that for some problem families, a per-coordinate learning rate for online gradient descent provides asymptotically less regret than even the best non-increasing global learning rate (chosen in hindsight, given the observed loss functions). This construction can be adapted to FTPRL as:

Theorem 1 *There exists a family of online convex optimization problems, parametrized by the number of rounds T , where online subgradient descent with a non-increasing learning rate sequence (and FTPRL with non-decreasing coordinate-constant regularization) incurs regret at least $\Omega(T^{\frac{2}{3}})$, whereas FTPRL with appropriate diagonal regularization matrices Q_t has regret $O(\sqrt{T})$.*

In fact, any online learning algorithm whose regret is $O(MD\sqrt{T})$ (where D is the L_2 diameter of the feasible region, and M is a bound on the L_2 norm of the gradients) will suffer regret $\Omega(T^{\frac{2}{3}})$ on this family of problems. Note that this does not contradict the $O(MD\sqrt{T})$ upper bound on the regret, because in this family of problems $D = T^{\frac{1}{6}}$ (and $M = 1$).

1.3 Adaptive algorithms and competitive ratios

In Section 3, we introduce specific schemes for selecting the regularization matrices Q_t for FTPRL, and show that for certain feasible sets, these algorithms provide bounds within a constant factor of those for the best post-hoc choice of matrices, namely

$$\inf_{\vec{Q}_T \in \mathcal{Q}^T} B_R(\vec{Q}_T, \vec{g}_T) \tag{4}$$

where $\mathcal{Q} \subseteq S_+^n$ is a set of allowed matrices; S_+^n is the set of symmetric positive semidefinite $n \times n$ matrices, with S_{++}^n the corresponding set of symmetric positive definite matrices. We consider three different choices for \mathcal{Q} : the set of coordinate-constant matrices $\mathcal{Q}_{\text{const}} = \{\alpha I \mid \alpha \geq 0\}$; the set of non-negative diagonal matrices,

$$\mathcal{Q}_{\text{diag}} = \{\text{diag}(\lambda_1, \dots, \lambda_n) \mid \lambda_i \geq 0\};$$

and, the full set of positive-semidefinite matrices, $\mathcal{Q}_{\text{full}} = S_+^n$.

We first consider the case where the feasible region is an L_p unit ball, namely $\mathcal{F} = \{x \mid \|x\|_p \leq 1\}$. For $p \in [1, 2]$, we show that a simple algorithm (an analogue of standard online gradient descent) that selects matrices from $\mathcal{Q}_{\text{const}}$ is $\sqrt{2}$ -competitive with the best post-hoc choice of matrices from the full set of positive semidefinite matrices $\mathcal{Q}_{\text{full}} = S_+^n$. This algorithm is presented in Corollary 8, and the competitive ratio is proved in Theorem 13.

In contrast to the result for $p \in [1, 2]$, we show that for L_p balls with $p > 2$ a coordinate-independent choice of matrices ($Q_t \in \mathcal{Q}_{\text{const}}$) does not in general obtain the post-hoc optimal bound (see Section 3.3), and hence per-coordinate adaptation can help. The benefit of per-coordinate adaptation is most pronounced for the L_∞ -ball, where the coordinates are essentially independent. In light of this, we develop an efficient algorithm (FTPRL-Diag, Algorithm 1) for adaptively selecting Q_t from $\mathcal{Q}_{\text{diag}}$, which uses scaling based on the width of \mathcal{F} in the coordinate directions (Corollary 9). In this corollary, we also show that this algorithm is $\sqrt{2}$ -competitive with the best post-hoc choice of matrices from $\mathcal{Q}_{\text{diag}}$ when the feasible set is a hyperrectangle.

While per-coordinate adaptation does not help for the unit L_2 -ball, it can help when the feasible set is a hyperellipsoid. In particular, in the case where $\mathcal{F} = \{x \mid \|Ax\|_2 \leq 1\}$ for $A \in S_{++}^n$, we show that an appropriate transformation of the problem can produce significantly better regret bounds. More generally, we show (see Theorem 12) that if one has a κ -competitive adaptive FTPRL scheme for the feasible set $\{x \mid \|x\| \leq 1\}$ for an arbitrary norm, it can be extended to provide a κ -competitive algorithm for feasible sets of the form $\{x \mid \|Ax\| \leq 1\}$. Using this result, we can show FTPRL-Scale is $\sqrt{2}$ -competitive with the best post-hoc choice of matrices from S_+^n when $\mathcal{F} = \{x \mid \|Ax\|_2 \leq 1\}$ and $A \in S_{++}^n$; it is $\sqrt{2}$ -competitive with $\mathcal{Q}_{\text{diag}}$ when $\mathcal{F} = \{x \mid \|Ax\|_p \leq 1\}$ for $p \in [1, 2]$.

Of course, in many practical applications the feasible set may not be so nicely characterized. We emphasize that our algorithms and analysis are applicable to arbitrary feasible sets, but the quality of the bounds and competitive ratios will depend on how tightly the feasible set can be approximated by a suitably chosen transformed norm ball. In Theorem 10, we show in particular that when FTPRL-Diag is applied to an arbitrary feasible set, it provides a competitive guarantee related to the ratio of the widths of the smallest hyperrectangle that contains \mathcal{F} to the largest hyperrectangle contained in \mathcal{F} .

1.4 Notation and technical background

We use the notation $g_{1:t}$ as a shorthand for $\sum_{\tau=1}^t g_\tau$. Similarly we write $Q_{1:t}$ for a sum of matrices Q_t , and $f_{1:t}$ to denote the function $f_{1:t}(x) = \sum_{\tau=1}^t f_\tau(x)$. We write $x^\top y$ or $x \cdot y$ for the inner product between $x, y \in \mathbb{R}^n$. The i th entry in a vector x is denoted $x_i \in \mathbb{R}$; when we have a sequence of vectors $x_t \in \mathbb{R}^n$ indexed by time, the i th entry is $x_{t,i} \in \mathbb{R}$. We use $\partial f(x)$ to denote the set of subgradients of f evaluated at x .

Recall $A \in S_{++}^n$ means $\forall x \neq 0, x^\top Ax > 0$. We use the generalized inequality $A \succ 0$ when $A \in S_{++}^n$, and similarly $A \prec B$ when $B - A \succ 0$, implying $x^\top Ax < x^\top Bx$. We define $A \preceq B$ analogously for symmetric positive semidefinite matrices S_+^n . For $B \in S_+^n$, we write $B^{1/2}$ for the square root of B , the unique $X \in S_+^n$ such that $XX = B$ (see, for example, Boyd and Vandenberghe, (2004, A.5.2)). We also make use of the fact that any $A \in S_+^n$ can be factored as $A = PDP^\top$ where $P^\top P = I$ and $D = \text{diag}(\lambda_1, \dots, \lambda_n)$ where λ_i are the eigenvalues of A .

Following the arguments of Zinkevich (2003), for the remainder we restrict our attention to linear functions. Briefly, the convexity of f_t implies $f_t(x) \geq g_t^\top(x - x_t) + f_t(x_t)$, where $g_t \in \partial f(x_t)$. Because this inequality is tight for $x = x_t$, it follows that regret measured against the affine functions on the right hand side is an upper bound on true regret. Furthermore, regret is unchanged if we replace this affine function with the linear function $g_t^\top x$. Thus, so long as our algorithm only makes use of the subgradients g_t , we may assume without loss of generality that the loss functions are linear.

Taking into account this reduction and the functional form of the r_t , the update of FTPRL is

$$x_{t+1} = \arg \min_{x \in \mathcal{F}} \left(\frac{1}{2} \sum_{\tau=1}^t (x - x_\tau)^\top Q_\tau (x - x_\tau) + g_{1:t} \cdot x \right). \quad (5)$$

2 Analysis of FTPRL

In this section, we prove the following bound on the regret of FTPRL for an arbitrary sequence of regularization matrices Q_t . In this section $\|\cdot\|$ always means the L_2 norm, $\|\cdot\|_2$.

Theorem 2 *Let $\mathcal{F} \subseteq \mathbb{R}^n$ be a closed, bounded convex set with $0 \in \mathcal{F}$. Let $Q_1 \in S_{++}^n$, and $Q_2, \dots, Q_T \in S_+^n$. Define $r_t(x) = \frac{1}{2} \|Q_t^{\frac{1}{2}}(x - x_t)\|_2^2$, and $A_t = (Q_{1:t})^{\frac{1}{2}}$. Let f_t be a sequence of loss functions, with $g_t \in \partial f_t(x_t)$ a sub-gradient of f_t at x_t . Then, the FTPRL algorithm that that faces loss functions f , plays $x_1 = 0$, and uses the update of Equation (5) thereafter, has a regret bound*

$$\text{Regret} \leq r_{1:T}(\hat{x}) + \sum_{t=1}^T \|A_t^{-1} g_t\|^2$$

where $\hat{x} = \arg \min_{x \in \mathcal{F}} f_{1:T}(x)$ is the post-hoc optimal feasible point.

To prove Theorem 2 we will make use of the following bound on the regret of FTRL, which holds for arbitrary (possibly non-convex) loss functions. This lemma can be proved along the lines of (Kalai & Vempala, 2005); for a complete proof see (McMahan & Streeter, 2010, Appendix A).

Lemma 3 *Let r_1, r_2, \dots, r_T be a sequence of non-negative functions. The regret of FTPRL (which plays x_t as defined by Equation (2)) is bounded by*

$$r_{1:T}(\hat{x}) + \sum_{t=1}^T (f_t(x_t) - f_t(x_{t+1}))$$

where \hat{x} is the post-hoc optimal feasible point.

Once Lemma 3 is established, to prove Theorem 2 it suffices to show that for all t ,

$$f_t(x_t) - f_t(x_{t+1}) \leq \|A_t^{-1} g_t\|^2. \quad (6)$$

To show this, we first establish an alternative characterization of our algorithm as solving an unconstrained optimization followed by a suitable projection onto the feasible set. Define the projection operator,

$$P_{\mathcal{F},A}(u) = \arg \min_{x \in \mathcal{F}} \|A(x - u)\|$$

We will show that the following is an equivalent formula for x_t :

$$\begin{aligned} u_{t+1} &= \arg \min_{u \in \mathbb{R}^n} (r_{1:t}(u) + g_{1:t} \cdot u) \\ x_{t+1} &= P_{\mathcal{F},A_t}(u_{t+1}). \end{aligned} \quad (7)$$

This characterization will be useful, because the unconstrained solutions depend only on the linear functions g_t , and the quadratic regularization, and hence are easy to manipulate in closed form.

To show this equivalence, first note that because $Q_t \in S_+^n$ is symmetric,

$$r_t(u) = \frac{1}{2}(u - x_t)^\top Q_t(u - x_t) = \frac{1}{2}u^\top Q_t u - x_t^\top Q_t u + \frac{1}{2}x_t^\top Q_t x_t.$$

Defining constants $q_t = Q_t x_t$ and $k_t = \frac{1}{2}x_t^\top Q_t x_t$, we can write

$$r_{1:t}(u) = \frac{1}{2}u^\top Q_{1:t}u - q_{1:t}u + k_{1:t}. \quad (8)$$

The equivalence is then a corollary of the following lemma, choosing $Q = Q_{1:t}$ and $h = g_{1:t} - q_{1:t}$ (note that the constant term $k_{1:t}$ does not influence the argmin).

Lemma 4 *Let $Q \in S_{++}^n$ and $h \in \mathbb{R}^n$, and consider the function*

$$f(x) = h^\top x + \frac{1}{2}x^\top Qx.$$

Let $\hat{u} = \arg \min_{u \in \mathbb{R}^n} f(u)$. Then, letting $A = Q^{\frac{1}{2}}$, we have $P_{\mathcal{F},A}(\hat{u}) = \arg \min_{x \in \mathcal{F}} f(x)$.

Proof: Note that $\nabla_u f(u) = h + Qu$, implying that $\hat{u} = -Q^{-1}h$. Consider the function

$$f'(x) = \frac{1}{2}\|Q^{\frac{1}{2}}(x - \hat{u})\|^2 = \frac{1}{2}(x - \hat{u})^\top Q(x - \hat{u}).$$

We have

$$\begin{aligned} f'(x) &= \frac{1}{2}(x^\top Qx - 2x^\top Q\hat{u} + \hat{u}^\top Q\hat{u}) && \text{(because } Q \text{ is symmetric)} \\ &= \frac{1}{2}(x^\top Qx + 2x^\top Q(Q)^{-1}h + \hat{u}^\top Q\hat{u}) \\ &= \frac{1}{2}(x^\top Qx + 2x^\top h + \hat{u}^\top Q\hat{u}) \\ &= f(x) + \frac{1}{2}\hat{u}^\top Q\hat{u}. \end{aligned}$$

Because $\frac{1}{2}\hat{u}^\top Q\hat{u}$ is constant with respect to x , it follows that

$$\arg \min_{x \in \mathcal{F}} f(x) = \arg \min_{x \in \mathcal{F}} f'(x) = P_{\mathcal{F},A}(\hat{u}),$$

where the last equality follows from the definition of the projection operator. ■

We now derive a closed-form solution to the unconstrained problem. It is easy to show $\nabla r_t(u) = Q_t u - Q_t x_t$, and so

$$\nabla r_{1:t}(u) = Q_{1:t}u - \sum_{\tau=1}^t Q_\tau x_\tau.$$

Because u_{t+1} is the optimum of the (strongly convex) unconstrained problem, and $r_{1:t}$ is differentiable, we must have $\nabla r_{1:t}(u_{t+1}) + g_{1:t} = 0$. Hence, we conclude $Q_{1:t}u_{t+1} - \sum_{\tau=1}^t Q_\tau x_\tau + g_{1:t} = 0$, or

$$u_{t+1} = Q_{1:t}^{-1} \left(\sum_{\tau=1}^t Q_\tau x_\tau - g_{1:t} \right). \quad (9)$$

This closed-form solution will let us bound the difference between u_t and u_{t+1} in terms of g_t . The next Lemma relates this distance to the difference between x_t and x_{t+1} , which determines our per round regret (Equation (6)). In particular, we show that the projection operator only makes u_t and u_{t+1} closer together, in terms of distance as measured by the norm $\|A_t \cdot\|$. We defer the proof to the end of the section.

Lemma 5 Let $Q \in S_{++}^n$ with $A = Q^{\frac{1}{2}}$. Let \mathcal{F} be a convex set, and let $u_1, u_2 \in \mathbb{R}^n$, with $x_1 = P_{\mathcal{F}, A}(u_1)$ and $x_2 = P_{\mathcal{F}, A}(u_2)$. Then,

$$\|A(x_2 - x_1)\| \leq \|A(u_1 - u_2)\|.$$

We now prove the following lemma, which will immediately yield the desired bound on $f_t(x_t) - f_t(x_{t+1})$.

Lemma 6 Let $Q \in S_{++}^n$ with $A = Q^{\frac{1}{2}}$. Let $v, g \in \mathbb{R}^n$, and let $u_1 = -Q^{-1}v$ and $u_2 = -Q^{-1}(v + g)$. Then, letting $x_1 = P_{\mathcal{F}, A}(u_1)$ and $x_2 = P_{\mathcal{F}, A}(u_2)$,

$$g^\top(x_1 - x_2) \leq \|A^{-1}g\|^2.$$

Proof: The fact that $Q = A^\top A \succ 0$ implies that $\|A \cdot\|$ and $\|A^{-1} \cdot\|$ are dual norms (see for example (Boyd & Vandenberghe, 2004, Sec. 9.4.1, pg. 476)). Using this fact,

$$\begin{aligned} g^\top(x_1 - x_2) &\leq \|A^{-1}g\| \cdot \|A(x_1 - x_2)\| \\ &\leq \|A^{-1}g\| \cdot \|A(u_1 - u_2)\| && \text{(Lemma 5)} \\ &= \|A^{-1}g\| \cdot \|A(Q^{-1}g)\| \\ &= \|A^{-1}g\| \cdot \|A(A^{-1}A^{-1}g)\| && \text{(Because } Q^{-1} = (AA)^{-1}\text{)} \\ &= \|A^{-1}g\| \cdot \|A^{-1}g\|. \end{aligned}$$

■

Proof of Theorem 2: First note that because $r_t(x_t) = 0$ and r_t is non-negative, $x_t = \arg \min_{x \in \mathcal{F}} r_t(x)$. For any functions f and g , if $x^* = \arg \min_{x \in \mathcal{F}} f(x)$ and $x^* = \arg \min_{x \in \mathcal{F}} g(x)$, then

$$x^* = \arg \min_{x \in \mathcal{F}} (f(x) + g(x)).$$

Thus we have

$$\begin{aligned} x_t &= \arg \min_{x \in \mathcal{F}} (g_{1:t-1}x + r_{1:t-1}(x)) \\ &= \arg \min_{x \in \mathcal{F}} (g_{1:t-1}x + r_{1:t}(x)) && \text{(Because } x_t = \arg \min_{x \in \mathcal{F}} r_t(x)\text{.)} \\ &= \arg \min_{x \in \mathcal{F}} \left(hx + \frac{1}{2}x^\top Q_{1:t}x \right) \end{aligned}$$

where the last line follows from Equation (8), letting $h = g_{1:t-1} - q_{1:t} = g_{1:t-1} - \sum_{\tau=1}^t Q_\tau x_\tau$, and dropping the constant $k_{1:t}$. For x_{t+1} , we have directly from the definitions

$$x_{t+1} = \arg \min_{x \in \mathcal{F}} (g_{1:t}x + r_{1:t}(x)) = \arg \min_{x \in \mathcal{F}} \left((h + g_t)x + \frac{1}{2}x^\top Q_{1:t}x \right).$$

Thus, Lemma 4 implies $x_t = P_{\mathcal{F}, A_t}(-(Q_{1:t})^{-1}h)$ and similarly $x_{t+1} = P_{\mathcal{F}, A_t}(-(Q_{1:t})^{-1}(h + g_t))$. Thus, by Lemma 6, $g_t(x_t - x_{t+1}) \leq \|A_t^{-1}g_t\|^2$. The theorem then follows from Lemma 3. ■

Proof of Lemma 5: Define

$$B(x, u) = \frac{1}{2}\|A(x - u)\|^2 = \frac{1}{2}(x - u)^\top Q(x - u),$$

so we can write equivalently

$$x_1 = \arg \min_{x \in \mathcal{F}} B(x, u_1).$$

Then, note that $\nabla_x B(x, u_1) = Qx - Qu_1$, and so we must have $(Qx_1 - Qu_1)^\top(x_2 - x_1) \geq 0$; otherwise for δ sufficiently small the point $x_1 + \delta(x_2 - x_1)$ would belong to \mathcal{F} (by convexity) and would be closer to u_1 than x_1 is. Similarly, we must have $(Qx_2 - Qu_2)^\top(x_1 - x_2) \geq 0$. Combining these, we have the following equivalent inequalities:

$$\begin{aligned} (Qx_1 - Qu_1)^\top(x_2 - x_1) - (Qx_2 - Qu_2)^\top(x_2 - x_1) &\geq 0 \\ (x_1 - u_1)^\top Q(x_2 - x_1) - (x_2 - u_2)^\top Q(x_2 - x_1) &\geq 0 \\ -(x_2 - x_1)^\top Q(x_2 - x_1) + (u_2 - u_1)^\top Q(x_2 - x_1) &\geq 0 \\ (u_2 - u_1)^\top Q(x_2 - x_1) &\geq (x_2 - x_1)^\top Q(x_2 - x_1). \end{aligned}$$

Letting $\hat{u} = u_2 - u_1$, and $\hat{x} = x_2 - x_1$, we have $\hat{x}^\top Q \hat{x} \leq \hat{u}^\top Q \hat{x}$. Since Q is positive semidefinite, we have $(\hat{u} - \hat{x})^\top Q (\hat{u} - \hat{x}) \geq 0$, or equivalently $\hat{u}^\top Q \hat{u} + \hat{x}^\top Q \hat{x} - 2\hat{x}^\top Q \hat{u} \geq 0$ (using the fact Q is also symmetric). Thus,

$$\hat{u}^\top Q \hat{u} \geq -\hat{x}^\top Q \hat{x} + 2\hat{x}^\top Q \hat{u} \geq -\hat{x}^\top Q \hat{x} + 2\hat{x}^\top Q \hat{x} = \hat{x}^\top Q \hat{x},$$

and so

$$\|A(u_2 - u_1)\|^2 = \hat{u}^\top Q \hat{u} \geq \hat{x}^\top Q \hat{x} = \|A(x_2 - x_1)\|^2. \quad \blacksquare$$

3 Specific Adaptive Algorithms and Competitive Ratios

Before proceeding to the specific results, we establish several results that will be useful in the subsequent arguments. In order to prove that adaptive schemes for selecting Q_t have good competitive ratios for the bound optimization problem, we will need to compare the bounds obtained by the adaptive scheme to the optimal post-hoc bound of Equation (4). Suppose the sequence Q_1, \dots, Q_T is optimal for Equation (4), and consider the alternative sequence $Q'_1 = Q_{1:T}$ and $Q'_t = 0$ for $t > 1$. Using the fact that $Q_{1:t} \succeq Q_{1:t-1}$ implies $Q_{1:t}^{-1} \preceq Q_{1:t-1}^{-1}$, it is easy to show the alternative sequence also achieves the minimum. It follows that a sequence with $Q_1 = Q$ on the first round, and $Q_t = 0$ thereafter is always optimal. Hence, to solve for the post-hoc bound we can solve an optimization of the form

$$\inf_{Q \in \mathcal{Q}} \left(\max_{\hat{y} \in \mathcal{F}_{\text{sym}}} \left(\frac{1}{2} \hat{y}^\top Q \hat{y} \right) + \sum_{t=1}^T g_t^\top Q^{-1} g_t \right). \quad (10)$$

The diameter of \mathcal{F} is $D \equiv \max_{y, y' \in \mathcal{F}} \|y - y'\|_2$, and so for $\hat{y} \in \mathcal{F}_{\text{sym}}$, $\|\hat{y}\|_2 \leq D$. When \mathcal{F} is symmetric ($x \in \mathcal{F}$ implies $-x \in \mathcal{F}$), we have $y \in \mathcal{F}$ if and only if $2y \in \mathcal{F}_{\text{sym}}$, so (10) is equivalent to:

$$\inf_{Q \in \mathcal{Q}} \left(\max_{y \in \mathcal{F}} (2y^\top Q y) + \sum_{t=1}^T g_t^\top Q^{-1} g_t \right). \quad (11)$$

For simplicity of exposition, we assume $g_{1,i} > 0$ for all i , which ensures that only positive definite matrices can be optimal.² This assumption also ensures $Q_1 \in S_{++}^n$ for the adaptive schemes discussed below, as required by Theorem 2. This is without loss of generality, as we can always hallucinate an initial loss function with arbitrarily small components, and this changes regret by an arbitrarily small amount. We will also use the following Lemma (Auer & Gentile, 2000):

Lemma 7 *For any non-negative real numbers x_1, x_2, \dots, x_n ,*

$$\sum_{i=1}^n \frac{x_i}{\sqrt{\sum_{j=1}^i x_j}} \leq 2\sqrt{\sum_{i=1}^n x_i}.$$

3.1 Adaptive coordinate-constant regularization

We derive bounds where Q_t is chosen from the set $\mathcal{Q}_{\text{const}}$, and show that this algorithm comes within a factor of $\sqrt{2}$ of using the best constant regularization strength λI . This algorithm achieves a bound of $\mathcal{O}(DM\sqrt{T})$ where D is the diameter of the feasible region and M is a bound on $\|g_t\|_2$, matching the best possible bounds in terms of these parameters (Abernethy et al., 2008). We will prove a much stronger competitive guarantee for this algorithm in Theorem 13.

Corollary 8 *Suppose \mathcal{F} has L_2 diameter D . Then, if we run FTPRL with diagonal matrices such that*

$$(Q_{1:t})_{ii} = \bar{\alpha}_t = \frac{2\sqrt{G_t}}{D}$$

where $G_t = \sum_{\tau=1}^t \sum_{i=1}^n g_{\tau,i}^2$, then

$$\text{Regret} \leq 2D\sqrt{G_T}.$$

If $\|g_t\|_2 \leq M$, then $G_T \leq M^2 T$, and this translates to a bound of $\mathcal{O}(DM\sqrt{T})$. When $\mathcal{F} = \{x \mid \|x\|_2 \leq D/2\}$, this bound is $\sqrt{2}$ -competitive for the bound optimization problem over $\mathcal{Q}_{\text{const}}$.

²In the case where \mathcal{F} has 0 width in some direction, the infimum will not be attained by a finite Q , but by a sequence that assigns 0 penalty (on the right-hand side) to the components of the gradient in the direction of 0 width, requiring some entries in Q to go to ∞ .

Algorithm 1 FTPRL-Diag

Input: feasible set $\mathcal{F} \subseteq \times_{i=1}^n [a_i, b_i]$
Initialize $x_1 = 0 \in \mathcal{F}$
 $(\forall i)$, $G_i = 0, q_i = 0, \lambda_{0,i} = 0, D_i = b_i - a_i$
for $t = 1$ **to** T **do**
 Play the point x_t , incur loss $f_t(x_t)$
 Let $g_t \in \partial f_t(x_t)$
 for $i = 1$ **to** n **do**
 $G_i = G_i + g_{t,i}^2$
 $\lambda_{t,i} = \frac{2}{D_i} \sqrt{G_i} - \lambda_{1:t-1,i}$
 $q_i = q_i + x_{t,i} \lambda_{t,i}$
 $u_{t+1,i} = (g_{1:t,i} - q_i) / \lambda_{1:t,i}$
 end for
 $A_t = \text{diag}(\sqrt{\lambda_{1:t,1}}, \dots, \sqrt{\lambda_{1:t,n}})$
 $x_{t+1} = \text{Project}_{\mathcal{F}, A_t}(u_{t+1})$
end for

Algorithm 2 FTPRL-Scale

Input: feasible set $\mathcal{F} \subseteq \{x \mid \|Ax\| \leq 1\}$,
with $A \in S_{++}^n$
Let $\hat{\mathcal{F}} = \{x \mid \|x\| \leq 1\}$
Initialize $x_1 = 0, (\forall i) D_i = b_i - a_i$
for $t = 1$ **to** T **do**
 Play the point x_t , incur loss $f_t(x_t)$
 Let $g_t \in \partial f_t(x_t)$
 $\hat{g}_t = (A^{-1})^\top g_t$
 $\bar{\alpha} = \sqrt{\sum_{\tau=1}^t \sum_{i=1}^n \hat{g}_{\tau,i}^2}$
 $\alpha_t = \bar{\alpha} - \alpha_{1:t-1}$
 $q_t = \alpha_t x_t$
 $\hat{u}_{t+1} = (1/\bar{\alpha})(q_{1:t} - g_{1:t})$
 $A_t = (\bar{\alpha}I)^{\frac{1}{2}}$
 $\hat{x}_{t+1} = \text{Project}_{\hat{\mathcal{F}}, A_t}(\hat{u}_{t+1})$
 $x_{t+1} = A^{-1} \hat{x}_{t+1}$
end for

Proof: Let the diagonal entries of Q_t all be $\alpha_t = \bar{\alpha}_t - \bar{\alpha}_{t-1}$ (with $\bar{\alpha}_0 = 0$), so $\alpha_{1:t} = \bar{\alpha}_t$. Note $\alpha_t \geq 0$, and so this choice is feasible. We consider the left and right-hand terms of Equation (3) separately. For the left-hand term, letting \hat{y}_t be an arbitrary sequence of points from \mathcal{F}_{sym} , and noting $\hat{y}_t^\top \hat{y}_t \leq \|\hat{y}_t\|_2 \cdot \|\hat{y}_t\|_2 \leq D^2$,

$$\frac{1}{2} \sum_{t=1}^T \hat{y}_t^\top Q_t \hat{y}_t = \frac{1}{2} \sum_{t=1}^T \hat{y}_t^\top \hat{y}_t \alpha_t \leq \frac{1}{2} D^2 \sum_{t=1}^T \alpha_t = \frac{1}{2} D^2 \bar{\alpha}_T = D \sqrt{G_T}.$$

For the right-hand term, we have

$$\sum_{t=1}^T g_t^\top Q_{1:t}^{-1} g_t = \sum_{t=1}^T \sum_{i=1}^n \frac{g_{t,i}^2}{\alpha_{1:t}} = \sum_{t=1}^T \frac{D \sum_{i=1}^n g_{t,i}^2}{\sqrt{G_t}} \leq D \sqrt{G_T},$$

where the last inequality follows from Lemma 7.

In order to make a competitive guarantee, we must prove a lower bound on the post-hoc optimal bound function B_R , Equation (10). This is in contrast to the upper bound that we must show for the regret of the algorithm. When $\mathcal{F} = \{x \mid \|x\|_2 \leq D/2\}$, Equation (10) simplifies to exactly

$$\min_{\alpha \geq 0} \left(\frac{1}{2} \alpha D^2 + \frac{1}{\alpha} G_T \right) = D \sqrt{2G_T} \quad (12)$$

and so we conclude the adaptive algorithm is $\sqrt{2}$ -competitive for the bound optimization problem. \blacksquare

3.2 Adaptive diagonal regularization

In this section, we introduce and analyze FTPRL-Diag, a specialization of FTPRL that uses regularization matrices from $\mathcal{Q}_{\text{diag}}$. Let $D_i = \max_{x, x' \in \mathcal{F}} |x_i - x'_i|$, the width of \mathcal{F} along the i th coordinate. We construct a bound on the regret of FTPRL-Diag in terms of these D_i . The D_i implicitly define a hyperrectangle that contains \mathcal{F} . When \mathcal{F} is in fact such a hyperrectangle, our bound is $\sqrt{2}$ -competitive with the best post-hoc optimal bound using matrices from $\mathcal{Q}_{\text{diag}}$.

Corollary 9 *Let \mathcal{F} be a convex feasible set of width D_i in coordinate i . We can construct diagonal matrices Q_t such that the i th entry on the diagonal of $Q_{1:t}$ is given by:*

$$\bar{\lambda}_{t,i} = \frac{2}{D_i} \sqrt{\sum_{\tau=1}^t g_{\tau,i}^2}.$$

Then the regret of FTPRL satisfies

$$\text{Regret} \leq 2 \sum_{i=1}^n D_i \sqrt{\sum_{t=1}^T g_{t,i}^2}.$$

When \mathcal{F} is a hyperrectangle, then this algorithm is $\sqrt{2}$ -competitive with the post-hoc optimal choice of Q_t from the $\mathcal{Q}_{\text{diag}}$. That is,

$$\text{Regret} \leq \sqrt{2} \inf_{Q \in \mathcal{Q}_{\text{diag}}} \left(\max_{\hat{y} \in \mathcal{F}_{\text{sym}}} \left(\frac{1}{2} \hat{y}^\top Q \hat{y} \right) + \sum_{t=1}^T g_t^\top Q^{-1} g_t \right).$$

Proof: The construction of $Q_{1:t}$ in the theorem statement implies $(Q_t)_{ii} = \lambda_{t,i} \equiv \bar{\lambda}_{t,i} - \bar{\lambda}_{t-1,i}$. These entries are guaranteed to be non-negative as $\bar{\lambda}_{t,i}$ is a non-decreasing function of t .

We begin from Equation (3), letting \hat{y}_t be an arbitrary sequence of points from \mathcal{F}_{sym} . For the left-hand term,

$$\frac{1}{2} \sum_{t=1}^T \hat{y}_t^\top Q_t \hat{y}_t = \frac{1}{2} \sum_{t=1}^T \sum_{i=1}^n \hat{y}_{t,i}^2 \lambda_{t,i} \leq \frac{1}{2} \sum_{i=1}^n D_i^2 \sum_{t=1}^T \lambda_{t,i} = \frac{1}{2} \sum_{i=1}^n D_i^2 \bar{\lambda}_{T,i} = \sum_{i=1}^n D_i \sqrt{\sum_{t=1}^T g_{t,i}^2}.$$

For the right-hand term, we have

$$\sum_{t=1}^T g_t^\top Q_{1:t}^{-1} g_t = \sum_{t=1}^T \sum_{i=1}^n \frac{g_{t,i}^2}{\bar{\lambda}_{t,i}} = \sum_{i=1}^n \frac{D_i}{2} \sum_{t=1}^T \frac{g_{t,i}^2}{\sqrt{\sum_{\tau=1}^t g_{\tau,i}^2}} \leq \sum_{i=1}^n D_i \sqrt{\sum_{t=1}^T g_{t,i}^2},$$

where the last inequality follows from Lemma 7. Summing these bounds on the two terms of Equation (3) yields the stated bound on regret.

Now, we consider the case where the feasible set is exactly a hyperrectangle, that is, $\mathcal{F} = \{x \mid x_i \in [a_i, b_i]\}$ where $D_i = b_i - a_i$. Then, the optimization of Equation (10) decomposes on a per-coordinate basis, and in particular there exists a $\hat{y} \in \mathcal{F}_{\text{sym}}$ so that $\hat{y}_i^2 = D_i^2$ in each coordinate. Thus, for $Q = \text{diag}(\lambda_1, \dots, \lambda_n)$, the bound function is exactly

$$\sum_{i=1}^n \frac{1}{2} \lambda_i D_i^2 + \frac{1}{\lambda_i} \sum_{t=1}^T g_{t,i}^2.$$

Choosing $\lambda_i = \frac{1}{D_i} \sqrt{2 \sum_{t=1}^T g_{t,i}^2}$ minimizes this quantity, producing a post-hoc bound of

$$\sqrt{2} \sum_{i=1}^n D_i \sqrt{\sum_{t=1}^T g_{t,i}^2},$$

verifying that the adaptive scheme is $\sqrt{2}$ -competitive with matrices from $\mathcal{Q}_{\text{diag}}$. ■

The regret guarantees of the FTPRL-Diag algorithm hold on arbitrary feasible sets, but the competitive guarantee only applies for hyperrectangles. We now extend this result, showing that a competitive guarantee can be made based on how well the feasible set is approximated by hyperrectangles.

Theorem 10 *Let \mathcal{F} be an arbitrary feasible set, bounded by a hyperrectangle H^{out} of width W_i in coordinate i ; further, let H^{in} be a hyperrectangle contained by \mathcal{F} , of width $w_i > 0$ in coordinate i . That is, $H^{\text{in}} \subseteq \mathcal{F} \subseteq H^{\text{out}}$. Let $\beta = \max_i \frac{W_i}{w_i}$. Then, the FTPRL-Diag is $\sqrt{2}\beta$ -competitive with $\mathcal{Q}_{\text{diag}}$ on \mathcal{F} .*

Proof: By Corollary 9, the adaptive algorithm achieves regret bounded by $2 \sum_{i=1}^n W_i \sqrt{\sum_{t=1}^T g_{t,i}^2}$. We now consider the best post-hoc bound achievable with diagonal matrices on \mathcal{F} . Considering Equation (10), it is clear that for any Q ,

$$\max_{y \in \mathcal{F}_{\text{sym}}} \frac{1}{2} y^\top Q y + \sum_{t=1}^T g_t^\top Q^{-1} g_t \geq \max_{y \in H_{\text{sym}}^{\text{in}}} \frac{1}{2} y^\top Q y + \sum_{t=1}^T g_t^\top Q^{-1} g_t,$$

since the feasible set for the maximization (\mathcal{F}_{sym}) is larger on the left-hand side. But, on the right-hand side we have the post-hoc bound for diagonal regularization on a hyperrectangle, which we computed in the previous section to be $\sqrt{2} \sum_{i=1}^n w_i \sqrt{\sum_{t=1}^T g_{t,i}^2}$. Because $w_i \geq \frac{W_i}{\beta}$ by assumption, this is lower bounded by $\frac{\sqrt{2}}{\beta} \sum_{i=1}^n W_i \sqrt{\sum_{t=1}^T g_{t,i}^2}$, which proves the theorem. ■

Having had success with L_∞ , we now consider the potential benefits of diagonal adaptation for other L_p -balls.

3.3 A post-hoc bound for diagonal regularization on L_p balls

Suppose the feasible set F is an unit L_p -ball, that is $F = \{x \mid \|x\|_p \leq 1\}$. We consider the post-hoc bound optimization problem of Equation (11) with $\mathcal{Q} = \mathcal{Q}_{\text{diag}}$. Our results are summarized in the following theorem.

Theorem 11 *For $p > 2$, the optimal regularization matrix for B_R in $\mathcal{Q}_{\text{diag}}$ is not coordinate-constant (i.e., not contained in $\mathcal{Q}_{\text{const}}$), except in the degenerate case where $G_i \equiv \sum_{t=1}^T g_{t,i}^2$ is the same for all i . However, for $p \leq 2$, the optimal regularization matrix in $\mathcal{Q}_{\text{diag}}$ always belongs to $\mathcal{Q}_{\text{const}}$.*

Proof: Since \mathcal{F} is symmetric, the optimal post-hoc choice will be in the form of Equation (11). Letting $Q = \text{diag}(\lambda_1, \dots, \lambda_n)$, we can re-write this optimization problem as

$$\max_{y: \|y\|_p \leq 1} \left(2 \sum_{i=1}^n y_i^2 \lambda_i \right) + \sum_{i=1}^n \frac{G_i}{\lambda_i}. \quad (13)$$

To determine the optimal λ vector, we first derive a closed form for the solution to the maximization problem on the left hand side, assuming $p \geq 2$ (we handle the case $p < 2$ separately below). First note that the inequality $\|y\|_p \leq 1$ is equivalent to $\sum_{i=1}^n |y_i|^p \leq 1$. Making the change of variable $z_i = y_i^2$, this is equivalent to $\sum_{i=1}^n z_i^{\frac{p}{2}} \leq 1$, which is equivalent to $\|z\|_{\frac{p}{2}} \leq 1$ (the assumption $p \geq 2$ ensures that $\|\cdot\|_{\frac{p}{2}}$ is a norm). Hence, the left-hand side optimization reduces to

$$\max_{z: \|z\|_{\frac{p}{2}} \leq 1} 2 \sum_{i=1}^n z_i \lambda_i = 2 \|\lambda\|_q,$$

where $q = \frac{p}{p-2}$, so that $\|\cdot\|_{\frac{p}{2}}$ and $\|\cdot\|_q$ are dual norms (allowing $q = \infty$ for $p = 2$). Thus, for $p \geq 2$, the above bound simplifies to

$$B(\lambda) = 2 \|\lambda\|_q + \sum_{i=1}^n \frac{G_i}{\lambda_i}. \quad (14)$$

First suppose $p > 2$, so that q is finite. Then, taking the gradient of $B(\lambda)$,

$$\nabla B(\lambda)_i = \frac{2}{q} \left(\sum_{i=1}^n \lambda_i^q \right)^{\frac{1}{q}-1} \cdot q \lambda_i^{q-1} - \frac{G_i}{\lambda_i^2} = 2 \left(\frac{\lambda_i}{\|\lambda\|_q} \right)^{q-1} - \frac{G_i}{\lambda_i^2},$$

using $\frac{1}{q} - 1 = -\frac{1}{q}(q-1)$. If we make all the λ_i 's equal (say, to λ_1), then for the left-hand side we get

$$\left(\frac{\lambda_i}{\|\lambda\|_q} \right)^{q-1} = \left(\frac{\lambda_1}{(n\lambda_1^{\frac{1}{q}})} \right)^{q-1} = \left(\frac{1}{n^{\frac{1}{q}}} \right)^{q-1} = n^{\frac{1}{q}-1}.$$

Thus the i^{th} component of the gradient is $2n^{\frac{1}{q}-1} - \frac{G_i}{\lambda_i^2}$, and so if not all the G_i 's are equal, some component of the gradient is non-zero. Because $B(\lambda)$ is differentiable and the $\lambda_i \geq 0$ constraints cannot be tight (recall $G_1 > 0$), this implies a constant λ_i cannot be optimal, hence the optimal regularization matrix is not in $\mathcal{Q}_{\text{const}}$.

For $p \in [1, 2]$, we show that the solution to Equation (13) is

$$B_{\infty}(\lambda) \equiv 2 \|\lambda\|_{\infty} + \sum_{i=1}^n \frac{G_i}{\lambda_i}. \quad (15)$$

For $p = 2$ this follows immediately from Equation (14), because when $p = 2$ we have $q = \infty$. For $p \in [1, 2)$, the solution to Equation (13) is at least $B_{\infty}(\lambda)$, because we can always set $y_i = 1$ for whatever λ_i is largest and set $y_j = 0$ for $j \neq i$. If $p < 2$ then the feasible set \mathcal{F} is a subset of the unit L_2 ball, so the solution to Equation (13) is upper bounded by the solution when $p = 2$, namely $B_{\infty}(\lambda)$. It follows that the solution is exactly $B_{\infty}(\lambda)$. Because the left-hand term of $B_{\infty}(\lambda)$ only penalizes for the largest λ_i , and on the right-hand we would like all λ_i as large as possible, a solution of the form $\lambda_1 = \lambda_2 = \dots = \lambda_n$ must be optimal. ■

3.4 Full matrix regularization on hyperspheres and hyperellipsoids

In this section, we develop an algorithm for feasible sets $\mathcal{F} \subseteq \{x \mid \|Ax\|_p \leq 1\}$, where $p \in [1, 2]$ and $A \in S_{++}^n$. When $\mathcal{F} = \{x \mid \|Ax\|_2 \leq 1\}$, this algorithm, FTPRL-Scale, is $\sqrt{2}$ -competitive with arbitrary $Q \in S_{++}^n$. For $\mathcal{F} = \{x \mid \|Ax\|_p \leq 1\}$ with $p \in [1, 2)$ it is $\sqrt{2}$ -competitive with Q_{diag} .

First, we show that rather than designing adaptive schemes specifically for linear transformations of norm balls, it is sufficient (from the point of view of analyzing FTPRL) to consider unit norm balls if suitable pre-processing is applied. In the same fashion that pre-conditioning may speed batch subgradient descent algorithms, we show this approach can produce significantly improved regret bounds when A is poorly conditioned (i.e., the ratio of the largest to smallest eigenvalue is large).

Theorem 12 *Fix an arbitrary norm $\|\cdot\|$, and define an online linear optimization problem $\mathcal{I} = (\mathcal{F}, (g_1, \dots, g_T))$ where $\mathcal{F} = \{x \mid \|Ax\| \leq 1\}$ with $A \in S_{++}^n$. We define the related instance $\hat{\mathcal{I}} = (\hat{\mathcal{F}}, (\hat{g}_1, \dots, \hat{g}_T))$, where $\hat{\mathcal{F}} = \{\hat{x} \mid \|\hat{x}\| \leq 1\}$ and $\hat{g}_t = A^{-1}g_t$. Then:*

- *If we run any algorithm dependent only on subgradients on $\hat{\mathcal{I}}$, and it plays $\hat{x}_1, \dots, \hat{x}_T$, then by playing the corresponding points $x_t = A^{-1}\hat{x}_t$ on \mathcal{I} we achieve identical loss and regret.*
- *The post-hoc optimal bound over arbitrary $Q \in S_{++}^n$ is identical for these two instances.*

Proof: First, we note that for any function h where $\min_{x: \|Ax\| \leq 1} h(x)$ exists,

$$\min_{x: \|Ax\| \leq 1} h(x) = \min_{\hat{x}: \|\hat{x}\| \leq 1} h(A^{-1}\hat{x}), \quad (16)$$

using the change of variable $\hat{x} = Ax$. For the first claim, note that $\hat{g}_t^\top = g_t^\top A^{-1}$, and so for all t , $\hat{g}_t^\top \hat{x}_t = g_t^\top A^{-1}Ax_t = g_t^\top x_t$, implying the losses suffered on $\hat{\mathcal{I}}$ and \mathcal{I} are identical. Applying Equation (16), we have

$$\min_{x: \|Ax\| \leq 1} g_{1:t}^\top x = \min_{\hat{x}: \|\hat{x}\| \leq 1} g_{1:t}^\top A^{-1}\hat{x} = \min_{\hat{x}: \|\hat{x}\| \leq 1} \hat{g}_{1:t}^\top \hat{x},$$

implying the post-hoc optimal feasible points for the two instances also incur identical loss. Combining these two facts proves the first claim. For the second claim, it is sufficient to show for any $Q \in S_{++}^n$ applied to the post-hoc bound for problem \mathcal{I} , there exists a $\hat{Q} \in S_{++}^n$ that achieves the same bound for $\hat{\mathcal{I}}$ (and vice versa). Consider such a Q for \mathcal{I} . Then, again applying Equation (16), we have

$$\max_{y: \|Ay\|_p \leq 1} (2y^\top Qy) + \sum_{t=1}^T g_t^\top Q^{-1}g_t = \max_{\hat{y}: \|\hat{y}\| \leq 1} (2\hat{y}^\top A^{-1}QA^{-1}\hat{y}) + \sum_{t=1}^T \hat{g}_t^\top AQA^{-1}A\hat{g}_t.$$

The left-hand side is the value of the post-hoc bound on \mathcal{I} from Equation (11). Noting that $(A^{-1}QA^{-1})^{-1} = AQA^{-1}A$, the right-hand side is the value of the post hoc bound for $\hat{\mathcal{I}}$ using $\hat{Q} = A^{-1}QA^{-1}$. The fact A^{-1} and Q are in S_{++}^n guarantees $\hat{Q} \in S_{++}^n$ as well, and the theorem follows. \blacksquare

We can now define the adaptive algorithm FTPRL-Scale: given a $\mathcal{F} \subseteq \{x \mid \|Ax\|_p \leq 1\}$, it uses the transformation suggested by Theorem 12, applying the coordinate-constant algorithm of Corollary 8 to the transformed instance, and playing the corresponding point mapped back into \mathcal{F} .³ Pseudocode is given as Algorithm 2.

Theorem 13 *The diagonal-constant algorithm analyzed in Corollary 8 is $\sqrt{2}$ -competitive with S_{++}^n when $\mathcal{F} = \{x \mid \|x\|_p \leq 1\}$ for $p = 2$, and $\sqrt{2}$ -competitive against Q_{diag} when $p \in [1, 2)$. Furthermore, when $\mathcal{F} = \{x \mid \|Ax\|_p \leq 1\}$ with $A \in S_{++}^n$, the FTPRL-Scale algorithm (Algorithm 2) achieves these same competitive guarantees. In particular, when $\mathcal{F} = \{x \mid \|x\|_2 \leq 1\}$, we have*

$$\text{Regret} \leq \sqrt{2} \inf_{Q \in S_{++}^n} \left(\max_{y \in \mathcal{F}} (2y^\top Qy) + \sum_{t=1}^T g_t^\top Q^{-1}g_t \right).$$

Proof: The results for Q_{diag} with $p \in [1, 2)$ follow from Theorems 11 and 12 and Corollary 8. We now consider the case $p = 2$. Consider a $Q \in S_{++}^n$ for Equation (11) (recall only a $Q \in S_{++}^n$ could be optimal since $g_1 > 0$). We can write $Q = PDP^\top$ where $D = \text{diag}(\lambda_1, \dots, \lambda_n)$ is a diagonal matrix of positive eigenvalues and $PP^\top = I$. It is then easy to verify $Q^{-1} = PD^{-1}P^\top$.

³By a slightly more cumbersome argument, it is possible to show that instead of applying this transformation, FTPRL can be run directly on \mathcal{F} using appropriately transformed Q_t matrices.

When $p = 2$ and $\mathcal{F} = \{x \mid \|x\|_p \leq 1\}$, Equation (15) is tight, and so the post-hoc bound for Q is

$$2 \max_i(\lambda_i) + \sum_{t=1}^T g_t^\top (PD^{-1}P^\top)g_t.$$

Let $z_t = P^\top g_t$, so each right-hand term is $\sum_{i=1}^n \frac{z_{t,i}^2}{\lambda_i}$. It is clear this quantity is minimized when each λ_i is chosen as large as possible, while on the left-hand side we are only penalized for the largest eigenvalue of Q (the largest λ_i). Thus, a solution where $D = \alpha I$ for $\alpha > 0$ is optimal. Plugging into the bound, we have

$$B(\alpha) = 2\alpha + \sum_{t=1}^T g_t^\top \left(P \left(\frac{1}{\alpha} I \right) P^\top \right) g_t = 2\alpha + \frac{1}{\alpha} \sum_{t=1}^T g_t^\top g_t = 2\alpha + \frac{G_T}{\alpha}$$

where we have used the fact that $PP^\top = I$. Setting $\alpha = \sqrt{G_T/2}$ produces a minimal post-hoc bound of $2\sqrt{2G_T}$. The diameter D is 2, so the coordinate-constant algorithm has regret bound $4\sqrt{G_T}$ (Corollary 8), proving the first claim of the theorem for $p = 2$. The second claim follows from Theorem 12. ■

Suppose we have a problem instance where $\mathcal{F} = \{x \mid \|Ax\|_2 \leq 1\}$ where $A = \text{diag}(1/a_1, \dots, 1/a_n)$ with $a_i > 0$. To demonstrate the advantage offered by this transformation, we can compare the regret bound obtained by directly applying the algorithm of Corollary 8 to that of the FTPRL-Scale algorithm. Assume WLOG that $\max_i a_i = 1$, implying the diameter of \mathcal{F} is 2. Let g_1, \dots, g_T be the loss functions for this instance. Recalling $G_i = \sum_{t=1}^T g_{t,i}^2$, applying Corollary 8 directly to this problem gives

$$\text{Regret} \leq 4 \sqrt{\sum_{i=1}^n G_i}. \quad (17)$$

This is the same as the bound obtained by online subgradient descent and related algorithms as well.

We now consider FTPRL-Scale, which uses the transformation of Theorem 12. Noting $D = 2$ for the hypersphere and applying Corollary 8 to the transformed problem gives an adaptive scheme with

$$\text{Regret} \leq 4 \sqrt{\sum_{i=1}^n \sum_{t=1}^T \hat{g}_{t,i}^2} = 4 \sqrt{\sum_{i=1}^n a_i^2 \sum_{t=1}^T g_{t,i}^2} = 4 \sqrt{\sum_{i=1}^n a_i^2 G_i}.$$

This bound is never worse than the bound of Equation (17), and can be arbitrarily better when many of the a_i are much smaller than 1.

4 Related work

In the batch convex optimization setting, it is well known that convergence rates can often be dramatically improved through the use of preconditioning, accomplished by an appropriate change of coordinates taking into account both the shape of the objective function and the feasible region (Boyd & Vandenberghe, 2004). To our knowledge, this is the first work that extends these concepts (necessarily in a quite different form) to the problem of online convex optimization, where they can provide a powerful tool for improving regret (the online analogue of convergence rates).

Perhaps the closest algorithms in spirit to our diagonal adaptation algorithm are confidence-weighted linear classification (Drezde et al., 2008) and AROW (Crammer et al., 2009), in that they make different-sized adjustments for different coordinates. Unlike our algorithm, these algorithms apply only to classification problems and not to general online convex optimization, and the guarantees are in the form of mistake bounds rather than regret bounds.

FTPRL is similar to the lazily-projected gradient descent algorithm of (Zinkevich, 2004, Sec. 5.2.3), but with a critical difference: the latter effectively centers regularization outside of the current feasible region (at u_t rather than x_t). As a consequence, lazily-projected gradient descent only attains low regret via a re-starting mechanism or a constant learning rate (chosen with knowledge of T). It is our technique of always centering additional regularization inside the feasible set that allows us to make guarantees for adaptively-chosen regularization.

Most recent state-of-the-art algorithms for online learning are in fact general algorithms for online convex optimization applied to learning problems. Many of these algorithms can be thought of as (significant) extensions of online subgradient descent, including (Duchi & Singer, 2009; Do et al., 2009; Shalev-Shwartz et al., 2007). Apart from the very general work of (Kalai & Vempala, 2005), few general follow-the-regularized-leader algorithms have been analyzed, with the notable exception of the recent work of Xiao (2009).

The notion of proving competitive ratios for regret bounds that are functions of regularization parameters is not unique to this paper. Bartlett et al. (2008) and Do et al. (2009) proved guarantees of this form, but for a different algorithm and class of regularization parameters.

In concurrent work (Streeter & McMahan, 2010), the authors proved bounds similar to those of Corollary 9 for online gradient descent with per-coordinate learning rates. These results were significantly less general than the ones presented here, and in particular were restricted to the case where \mathcal{F} was exactly a hyperrectangle. The FTPRL algorithm and bounds proved in this paper hold for arbitrary feasible sets, with the bound depending on the shape of the feasible set as well as the width along each dimension. Some results similar to those in this work were developed concurrently by Duchi et al. (2010), though for a different algorithm and using different analysis techniques.

5 Conclusions

In this work, we analyzed a new algorithm for online convex optimization, which takes ideas both from online subgradient descent as well as follow-the-regularized-leader. In our analysis of this algorithm, we show that the learning rates that occur in standard bounds can be replaced by positive semidefinite matrices. The extra degrees of freedom offered by these generalized learning rates provide the key to proving better regret bounds. We characterized the types of feasible sets where this technique can lead to significant gains, and showed that while it does not help on the hypersphere, it can have dramatic impact when the feasible set is a hyperrectangle.

The diagonal adaptation algorithm we introduced can be viewed as an incremental optimization of the formula for the final bound on regret. In the case where the feasible set really is a hyperrectangle, this allows us to guarantee our final regret bound is within a small constant factor of the best bound that could have been obtained had the full problem been known in advance. The diagonal adaptation algorithm is efficient, and exploits exactly the kind of structure that is typical in large-scale real-world learning problems such as click-through rate prediction and text classification.

Our work leaves open a number of interesting directions for future work, in particular the development of competitive algorithms for arbitrary feasible sets (without resorting to bounding norm-balls), and the development of algorithms that optimize over richer families of regularization functions.

References

- Abernethy, J., Bartlett, P. L., Rakhlin, A., & Tewari, A. (2008). Optimal strategies and minimax lower bounds for online convex games. *COLT*.
- Auer, P., & Gentile, C. (2000). Adaptive and self-confident on-line learning algorithms. *COLT*.
- Bartlett, P. L., Hazan, E., & Rakhlin, A. (2008). Adaptive online gradient descent. *NIPS*.
- Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. *NIPS*.
- Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. New York, NY, USA.
- Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press.
- Crammer, K., Kulesza, A., & Drezde, M. (2009). Adaptive regularization of weight vectors. *NIPS*.
- Do, C. B., Le, Q. V., & Foo, C.-S. (2009). Proximal regularization for online and batch learning. *ICML*.
- Drezde, M., Crammer, K., & Pereira, F. (2008). Confidence-weighted linear classification. *ICML*.
- Duchi, J., Hazan, E., & Singer, Y. (2010). Adaptive subgradient methods for online learning and stochastic optimization. Manuscript.
- Duchi, J., & Singer, Y. (2009). Efficient learning using forward-backward splitting. *NIPS*.
- Kalai, A., & Vempala, S. (2005). Efficient algorithms for online decision problems. *Journal of Computer and Systems Sciences*, 71.
- McMahan, H. B., & Streeter, M. (2010). Adaptive bound optimization for online convex optimization (extended version). <http://arxiv.org/abs/1002.4908>.
- Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal Estimated sub-GrAdient SOLver for SVM. *ICML*.
- Streeter, M., & McMahan, H. B. (2010). Less regret via online conditioning. <http://arxiv.org/abs/1002.4862>.
- Xiao, L. (2009). Dual averaging method for regularized stochastic learning and online optimization. *NIPS*.
- Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. *ICML*.
- Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *ICML*.
- Zinkevich, M. (2004). *Theoretical guarantees for algorithms in multi-agent settings*. Doctoral dissertation, Pittsburgh, PA, USA.

Adaptive Subgradient Methods for Online Learning and Stochastic Optimization

John Duchi
University of California, Berkeley
jduchi@cs.berkeley.edu

Elad Hazan
IBM Almaden Research Center
ehazan@cs.princeton.edu

Yoram Singer
Google Research
singer@google.com

Abstract

We present a new family of subgradient methods that dynamically incorporate knowledge of the geometry of the data observed in earlier iterations to perform more informative gradient-based learning. The adaptation, in essence, allows us to find needles in haystacks in the form of very predictive yet rarely observed features. Our paradigm stems from recent advances in online learning which employ proximal functions to control the gradient steps of the algorithm. We describe and analyze an apparatus for adaptively modifying the proximal function, which significantly simplifies the task of setting a learning rate and results in regret guarantees that are provably as good as the best proximal function that can be chosen in hindsight. We corroborate our theoretical results with experiments on a text classification task, showing substantial improvements for classification with sparse datasets.

1 Introduction

In many applications of online and stochastic learning, the input instances are of very high dimension, yet within any particular instance only a few features are non-zero. It is often the case, however, that the infrequently occurring features are highly informative and discriminative. The informativeness of rare features has led practitioners to craft domain-specific feature weightings, such as TF-IDF (Salton and Buckley, 1988), which pre-emphasize infrequently occurring features. We use this old idea as a motivation for applying modern learning-theoretic techniques to the problem of online and stochastic learning, focusing specifically on (sub)gradient methods.

Standard stochastic subgradient methods largely follow a predetermined procedural scheme that is oblivious to the characteristics of the data being observed. In contrast, our algorithms dynamically incorporate knowledge of the geometry of the data from earlier iterations to perform more informative gradient-based learning. Informally, our procedures associate frequently occurring features with low learning rates and infrequent features high learning rates. This construction prompts the learner to “take notice” each time an infrequent feature is observed. Thus, the adaptation facilitates identification and adaptation of highly predictive but comparatively rare features.

1.1 The Adaptive Gradient Algorithm

For simplicity, consider the basic online convex optimization setting. The algorithm iteratively makes a prediction $x_t \in \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is a closed convex set, and then receives a convex loss function f_t . Define the regret with respect to the (optimal) predictor $x^* \in \mathcal{X}$ as

$$R(T) \triangleq \sum_{t=1}^T [f_t(x_t) - f_t(x^*)] .$$

To achieve low regret, standard subgradient algorithms move the predictor x_t in the opposite direction of the subgradient $g_t \in \partial f_t(x_t)$ of the loss via the projected gradient update (e.g. Zinkevich, 2003)

$$x_{t+1} = \Pi_{\mathcal{X}}(x_t - \eta g_t) .$$

Our algorithm, called ADAGRAD, makes a second-order correction to the predictor using the previous loss functions. Denote the projection of a point y onto \mathcal{X} by $\Pi_{\mathcal{X}}^A(y) = \operatorname{argmin}_{x \in \mathcal{X}} \|x - y\|_A$ (where $\|x\|_A = \sqrt{\langle x, Ax \rangle}$). In this notation, our adaptation of gradient descent employs the update

$$x_{t+1} = \Pi_{\mathcal{X}}^{G_t^{1/2}}(x_t - \eta G_t^{-1/2} g_t) , \tag{1}$$

where the matrix $G_t = \sum_{\tau=1}^t g_\tau g_\tau^\top$ is the outer product of all previous subgradients. The above algorithm may be computationally impractical in high dimensions since it requires computation of the matrix square root of G_t , the outer product matrix. We therefore also analyze a version in which we use $\text{diag}(G_t)$, the diagonal of the outer product matrix, instead of G_t :

$$x_{t+1} = \Pi_{\mathcal{X}}^{\text{diag}(G_t)^{1/2}} \left(x_t - \eta \text{diag}(G_t)^{-1/2} g_t \right). \quad (2)$$

This latter update rule can be computed in linear time. Moreover, as we discuss later, when the vectors g_t are sparse the update can often be performed in time proportional to the support of the gradient.

Let us compare the regret bounds attained by both variants of gradient descent. Let the diameter of \mathcal{X} be bounded, so $\sup_{x,y \in \mathcal{X}} \|x - y\|_2 \leq D_2$. Then Zinkevich's analysis of online gradient descent—with the optimal choice in *hindsight* for the stepsize η —achieves regret

$$R(T) \leq \sqrt{2} D_2 \sqrt{\sum_{t=1}^T \|g_t\|_2^2}. \quad (3)$$

When \mathcal{X} is bounded via $\sup_{x,y \in \mathcal{X}} \|x - y\|_\infty \leq D_\infty$, the following corollary is a consequence of our main Theorem 5.

Corollary 1 *Let the sequence $\{x_t\} \subset \mathbb{R}^d$ be generated by the update in Eq. (6) and let $\max_t \|x^* - x_t\|_\infty \leq D_\infty$. Then with stepsize $\eta = D_\infty/\sqrt{2}$, for any x^* ,*

$$R(T) \leq \sqrt{2d} D_\infty \sqrt{\inf_{s \geq 0, \langle 1, s \rangle \leq d} \sum_{t=1}^T \|g_t\|_{\text{diag}(s)^{-1}}^2} = \sqrt{2} D_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2.$$

The important parts of the bound are the infimum under the root, which allows us to perform better than using the identity matrix, and the fact that the stepsize is easy to set a priori. For example, if $\mathcal{X} = \{x : \|x\|_\infty \leq 1\}$, then $D_2 = 2\sqrt{d}$ while $D_\infty = 2$. In the case of learning a dense predictor over a box, the bound in Corollary 1 is thus better than Eq. (3) as the identity matrix belongs to the set over which we take the infimum.

1.2 Improvement and Motivating Examples

In Section 6, we give empirical evidence in favor of adaptive algorithms. Here we give a few theoretical examples that show that for sparse data—input sequences where g_t has low cardinality—the adaptive methods are likely to perform better than non-adaptive methods. In all the cases we consider in this section we use the hinge loss, $f_t(x) = [1 - y_t \langle z_t, x \rangle]_+$, where y_t is the label of example t and $z_t \in \mathbb{R}^d$ is a data vector.

To begin, consider the following example of sparse random data. Assume that at each round t , feature i appears with probability $p_i = \min\{1, ci^{-\alpha}\}$ for some $\alpha \geq 2$ and a constant c . Suppose also that with probability 1, at least one feature appears, for instance by setting $p = 1$. Taking the expectation of the bound in Corollary 1, we have

$$\mathbb{E} \sum_{i=1}^d \|g_{1:T,i}\|_2 = \sum_{i=1}^d \mathbb{E} \sqrt{|\{t : |g_{t,i}| = 1\}|} \leq \sum_{i=1}^d \sqrt{\mathbb{E} |\{t : |g_{t,i}| = 1\}|} = \sum_{i=1}^d \sqrt{p_i T}$$

where to obtain the inequality above we used Jensen's inequality. Now, notice that for the rightmost sum, we have $c \sum_{i=1}^d i^{-\alpha/2} = O(\log d)$ since $\alpha \geq 2$. If the domain is a hypercube, $\mathcal{X} = \{x : \|x\|_\infty \leq 1\}$, then $D_\infty = 2$. Thus, the regret bound of ADAGRAD is $R(T) = O(\log d \sqrt{T})$. In contrast, the standard regret bound from Eq. (3) has $D_2 = 2\sqrt{d}$, and we know that $\|g_t\|_2^2 \geq 1$, yielding a regret bound $R(T) = O(\sqrt{dT})$.¹ Thus, ADAGRAD's regret guarantee is exponentially smaller than the non-adaptive regret bound as a function of dimension for this sparse data setting.

Next we give two concrete examples for which the adaptive methods learn a perfect predictor after d iterations, while standard online gradient descent (Zinkevich, 2003) suffers much higher loss. We assume the domain \mathcal{X} is compact and thus for online gradient descent we set $\eta_t = 1/\sqrt{t}$, which gives $O(\sqrt{T})$ regret.

Diagonal Adaptation In this first example, we consider the diagonal version of our proposed update in Eq. (2) with $\mathcal{X} = \{x : \|x\|_\infty \leq 1\}$. Evidently, this choice results in the update $x_{t+1} = x_t - \eta \text{diag}(G_t)^{-1/2} g_t$ followed by projection onto \mathcal{X} . Let e_i denote the i th unit basis vector, and assume that for each t , $z_t = \pm e_i$ for some i . Also let $y_t = \text{sign}(\langle \mathbb{1}, z_t \rangle)$ so that there exists a perfect classifier $x^* = \mathbb{1} \in \mathcal{X}$. We initialize x_1

¹ For $\alpha \in (1, 2)$, ADAGRAD has regret $R(T) = O(d^{1-\alpha/2} \sqrt{T}) = o(\sqrt{dT})$.

to be the zero vector. On rounds $t = 1, \dots, d$, we set $z_t = \pm e_t$, selecting the sign at random. It is clear that both diagonal adaptive descent and online gradient descent suffer a unit loss on each of the first d examples. However, the updates to parameter x_i on iteration i differ and amount to

$$x_{t+1} = x_t + e_t \quad (\text{ADAGRAD}) \quad x_{t+1} = x_t + \frac{1}{\sqrt{t}} e_t \quad (\text{Gradient Descent}).$$

After the first d rounds, the adaptive predictor has $x_{d+1} = x_{d+\tau} = \mathbb{1}$ for all $\tau \geq 1$ and suffers no further losses. The magnitude of the majority of the coordinates for gradient descent, though, is bounded by $\sum_{i=1}^t \frac{1}{\sqrt{d/2+id}} \leq \frac{2\sqrt{t}}{\sqrt{d}}$ after td iterations. Hence, for $\Omega(\sqrt{d})$ iterations, the loss suffered per coordinate is bounded from zero, for a total loss of $\Omega(d\sqrt{d})$ (compared with $O(d)$ for ADAGRAD). With larger stepsizes η/\sqrt{t} , gradient descent may suffer lower loss; however, an adversary can play $z_t = e_1$ indefinitely, forcing online gradient descent to suffer $\Omega(d^2)$ loss while ADAGRAD suffers constant regret per dimension.

Full Matrix Adaptation The above construction applies to the full matrix algorithm of Eq. (1) as well, but in more general scenarios, as per the following example. When using full matrix proximal functions we set $\mathcal{X} = \{x : \|x\|_2 \leq \sqrt{d}\}$. Let $V = [v_1 \dots v_d] \in \mathbb{R}^{d \times d}$ be an orthonormal matrix. Instead of z_t cycling through the unit vectors, we have z_t cycle through the v_i so that $z_t = \pm v_{(t \bmod d)+1}$. We let the label $y_t = \text{sign}(\langle \mathbb{1}, V^\top z_t \rangle) = \text{sign}(\sum_{i=1}^d \langle v_i, z_t \rangle)$. We provide an elaborated explanation in the full version of this paper (Duchi et al., 2010a). Intuitively, ADAGRAD needs to observe each orthonormal vector v_i only once while stochastic gradient descent's loss is again $\Omega(d\sqrt{d})$.

1.3 Framework and Outline of Results

Before describing our results formally, let us establish notation. Vectors and scalars are lower case italic letters, such as $x \in \mathcal{X}$. We denote a sequence of vectors by subscripts, i.e. x_t, x_{t+1}, \dots , and entries of each vector by an additional subscript, e.g. $x_{t,j}$. The subdifferential set of a function f evaluated at x is denoted $\partial f(x)$, and a particular vector in the subdifferential set is denoted by $f'(x) \in \partial f(x)$ or $g_t \in \partial f_t(x_t)$. We use $\langle x, y \rangle$ to denote the inner product between x and y . The Bregman divergence associated with a strongly convex and differentiable function ψ is

$$B_\psi(x, y) = \psi(x) - \psi(y) - \langle \nabla \psi(y), x - y \rangle.$$

For a matrix $A \in \mathbb{R}^{d \times d}$, $\text{diag}(A) \in \mathbb{R}^d$ denotes its diagonal, while for a vector $s \in \mathbb{R}^d$, $\text{diag}(s)$ denotes the diagonal matrix with s as its diagonal. We also make frequent use of the following two matrices. Let $g_{1:t} = [g_1 \dots g_t]$ denote the matrix obtained by concatenating the subgradient sequence. We denote the i th row of this matrix, which amounts to the concatenation of the i th component of each subgradient we observe, by $g_{1:t,i}$. Lastly, we define the outer product matrix $G_t = \sum_{\tau=1}^t g_\tau g_\tau^\top$.

We describe and analyze several different online learning algorithms and their stochastic convex optimization counterparts. Formally, we consider online learning with a sequence of composite functions ϕ_t . Each function is of the form $\phi_t(x) = f_t(x) + \varphi(x)$ where f_t and φ are (closed) convex functions. In the learning settings we study, f_t is either an instantaneous loss or a stochastic estimate of the objective function. The function φ serves as a fixed regularization function and is typically used to control the complexity of x . At each round the algorithm makes a prediction $x_t \in \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is a closed convex set, and then receives the function f_t . We define the regret with respect to the (optimal) predictor $x^* \in \mathcal{X}$ as

$$R_\phi(T) \triangleq \sum_{t=1}^T [\phi_t(x_t) - \phi_t(x^*)] = \sum_{t=1}^T [f_t(x_t) + \varphi(x_t) - f_t(x^*) - \varphi(x^*)]. \quad (4)$$

Our analysis applies to multiple methods for minimizing the regret defined in Eq. (4). The first is Nesterov's primal-dual subgradient method (Nesterov, 2009), and in particular Xiao's 2009 extension, regularized dual averaging (RDA) (Xiao, 2009), and the follow-the-regularized-leader (FTRL) family of algorithms (e.g. Kalai and Vempala, 2003; Hazan et al., 2006). In the primal-dual subgradient method the algorithm makes a prediction x_t on round t using the average gradient $\bar{g}_t = \frac{1}{t} \sum_{\tau=1}^t g_\tau$. The update encompasses a trade-off between a gradient-dependent linear term, the regularizer φ , and a strongly-convex term ψ_t for well-conditioned predictions. Here ψ_t is the proximal term. The update amounts to solving the problem

$$x_{t+1} = \underset{x \in \mathcal{X}}{\text{argmin}} \left\{ \eta \langle \bar{g}_t, x \rangle + \eta \varphi(x) + \frac{1}{t} \psi_t(x) \right\}, \quad (5)$$

where η is a step-size. The second method also has many names, such as proximal gradient, forward-backward splitting, and composite mirror descent (Tseng, 2008; Duchi and Singer, 2009; Duchi et al., 2010b).

We use the term composite mirror descent. The composite mirror descent method employs a more immediate trade-off between the current gradient g_t , φ , and staying close to x_t using the proximal function ψ ,

$$x_{t+1} = \operatorname{argmin}_{x \in \mathcal{X}} \{ \eta \langle g_t, x \rangle + \eta \varphi(x) + B_{\psi_t}(x, x_t) \} . \quad (6)$$

Our work focuses on temporal adaptation of the proximal function in a data driven way, while previous work simply sets $\psi_t \equiv \psi$, $\psi_t(\cdot) = \sqrt{t}\psi(\cdot)$, or $\psi_t(\cdot) = t\psi(\cdot)$ for some fixed ψ .

We provide formal analyses equally applicable to the above two updates and show how to automatically choose the function ψ_t so as to achieve asymptotically small regret. We describe and analyze two algorithms. Both algorithms use squared Mahalanobis norms as their proximal functions, setting $\psi_t(x) = \frac{1}{2} \langle x, H_t x \rangle$ for a symmetric matrix $H_t \succeq 0$. The first uses diagonal matrices while the second constructs full dimensional matrices. Concretely, we set

$$H_t = \operatorname{diag}(G_t)^{1/2} \text{ (Diagonal)} \quad \text{and} \quad H_t = G_t^{1/2} \text{ (Full)} . \quad (7)$$

Plugging the appropriate matrix from the above equation into ψ_t in Eq. (5) or Eq. (6) gives rise to our ADAGRAD family of algorithms. Informally, we obtain algorithms similar to second-order gradient descent by constructing approximations to the Hessian of the functions f_t . These approximations are conservative since we rely on the root of the gradient matrices.

We now outline our results, deferring formal statements of the theorems to later sections. Recall the definitions of $g_{1:t}$ as the matrix of concatenated subgradients and G_t as the outer product matrix in the prequel. When the proximal function $\psi_t(x) = \langle x, \operatorname{diag}(G_t)^{1/2} x \rangle$, the ADAGRAD algorithm has bounds attainable in time at most linear in the dimension d of the problem of

$$R_\phi(T) = O\left(\|x^*\|_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2\right) \quad \text{and} \quad R_\phi(T) = O\left(\max_{t \leq T} \|x_t - x^*\|_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2\right).$$

We also show that

$$\sum_{i=1}^d \|g_{1:T,i}\|_2 = d^{1/2} \sqrt{\inf_s \left\{ \sum_{t=1}^T \langle g_t, \operatorname{diag}(s)^{-1} g_t \rangle : s \succeq 0, \langle \mathbb{1}, s \rangle \leq d \right\}} .$$

The ADAGRAD algorithm with full matrix divergences entertains bounds of the form

$$R_\phi(T) = O\left(\|x^*\|_2 \operatorname{tr}(G_T^{1/2})\right) \quad \text{and} \quad R_\phi(T) = O\left(\max_{t \leq T} \|x_t - x^*\|_2 \operatorname{tr}(G_T^{1/2})\right).$$

Similar to the diagonal proximal function case, we further show that

$$\operatorname{tr}\left(G_T^{1/2}\right) = d^{1/2} \sqrt{\inf_S \left\{ \sum_{t=1}^T \langle g_t, S^{-1} g_t \rangle : S \succeq 0, \operatorname{tr}(S) \leq d \right\}} .$$

We formally state the above regret bounds in Theorems 5 and 8, respectively, and we give further discussion in their corollaries. Essentially, the theorems give oracle inequalities for online optimization. Though the specific sequence of gradients g_t received by the algorithm changes when there is adaptation, the inequalities say that our regret bounds are as good as the best quadratic proximal function in hindsight.

1.4 Related Work

The idea of adaptation in first order (gradient) methods is by no means new and can be traced back at least to the 1970s. There, we find [Shor's](#) work on space dilation methods (1972) as well as variable metric methods, such as the BFGS family of algorithms (e.g. [Fletcher, 1970](#)). This older work usually assumes that the function to be minimized is differentiable and, to our knowledge, did not consider stochastic, online, or composite optimization. More recently, [Bordes et al. \(2009\)](#) proposed carefully designed Quasi-Newton stochastic gradient descent, which is similar in spirit to our methods. However, their convergence results assume a smooth objective function whose Hessian is strictly positive definite and bounded away from 0. Our results are applicable in more general settings. In the online learning literature, there are results on adaptively choosing a learning rate η_t based on data seen so far ([Auer et al., 2002](#); [Bartlett et al., 2007](#)). We, in contrast, actively adapt the proximal function ψ itself.

The framework that is most related to ours is probably confidence weighted learning, whose most recent success is the adaptive regularization of weights algorithm (AROW) of [Crammer et al. \(2009\)](#). [Crammer et al.](#) give a mistake-bound analysis for online binary classification, which is similar in spirit to the second-order

Perceptron (Cesa-Bianchi et al., 2005). AROW maintains a mean prediction vector $\mu_t \in \mathbb{R}^d$ and a covariance matrix $\Sigma_t \in \mathbb{R}^{d \times d}$ over μ_t as well. At every step of the algorithm, the learner receives a pair (z_t, y_t) where $z_t \in \mathbb{R}^d$ is the t th example and $y_t \in \{-1, +1\}$ is the label. Whenever the predictor μ_t has margin less than 1, AROW performs the update

$$\beta_t = \frac{1}{\langle z_t, \Sigma_t z_t \rangle + \lambda}, \quad \alpha_t = [1 - y_t \langle z_t, \mu_t \rangle]_+, \quad \mu_{t+1} = \mu_t + \alpha_t \Sigma_t y_t z_t, \quad \Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t x_t x_t^\top \Sigma_t. \quad (8)$$

In the above, one can set Σ_t to be diagonal, which reduces run-time and storage requirements but still gives good performance (Crammer et al., 2009). In contrast to AROW, the ADAGRAD family uses the *root* of a covariance-like matrix, a consequence of our formal analysis. Crammer et al.’s algorithm and our algorithms have similar run times—linear in the dimension d —when using diagonal matrices. However, when using full matrices the runtime of their algorithm is $O(d^2)$, which is faster than ours.

Our approach differs from previous approaches since instead of focusing on a particular loss function or mistake bound, we view the problem of adapting the proximal function as an online (meta) learning problem. We then obtain bounds comparable to the bound obtained using the best proximal function chosen in hindsight. Our bounds are applicable to any convex Lipschitz loss and composite objective functions.

2 Adaptive Proximal Functions

In this section we give the template regret bounds for the family of subgradient algorithms we consider. Examining several well-known optimization bounds (e.g. Beck and Teboulle, 2003; Nesterov, 2009; Duchi et al., 2010b), we see that we can bound the regret as

$$R_\phi(T) \leq \frac{1}{\eta} B_\psi(x^*, x_1) + \frac{\eta}{2} \sum_{t=1}^T \|f'_t(x_t)\|_*^2. \quad (9)$$

Most of the regret depends on dual-norms of $f'_t(x_t)$, where the dual norm in turn depends on the choice of ψ . This naturally leads to the question of whether we can modify the proximal term ψ along the run of the algorithm in order to lower the contribution of the aforementioned norms. We achieve this goal by keeping second order information about the sequence f_t .

We begin by providing two corollaries based on previous work that give the regret of our base algorithms when the proximal function ψ_t is allowed to change. We assume that ψ_t is monotonically non-decreasing, that is, $\psi_{t+1}(x) \geq \psi_t(x)$. We also assume that ψ_t is 1-strongly convex with respect to a time-dependent seminorm $\|\cdot\|_{\psi_t}$. Formally,

$$\psi_t(y) \geq \psi_t(x) + \langle \nabla \psi_t(x), y - x \rangle + \frac{1}{2} \|x - y\|_{\psi_t}^2.$$

Strong convexity is guaranteed if and only if $B_{\psi_t}(x, y) \geq \frac{1}{2} \|x - y\|_{\psi_t}^2$. We also denote the dual norm of $\|\cdot\|_{\psi_t}$ by $\|\cdot\|_{\psi_t^*}$. For completeness, we provide the proofs of following two corollaries in the long version of this paper (Duchi et al., 2010a), though they build straightforwardly on Duchi et al. (2010b) and Xiao (2009). For the primal-dual subgradient update of Eq. (5), the following regret bound holds.

Corollary 2 *Let the sequence $\{x_t\}$ be defined by the update in Eq. (5). Then for any x^* , we have*

$$R_\phi(T) \leq \frac{1}{\eta} \psi_T(x^*) + \frac{\eta}{2} \sum_{t=1}^T \|f'_t(x_t)\|_{\psi_{t-1}^*}^2. \quad (10)$$

For composite mirror descent algorithms (Eq. (6)), under the assumption w.l.o.g. that $\varphi(x_1) = 0$, we have

Corollary 3 *Let the sequence $\{x_t\}$ be defined by the update in Eq. (6). Then for any x^* ,*

$$R_\phi(T) \leq \frac{1}{\eta} B_{\psi_1}(x^*, x_1) + \frac{1}{\eta} \sum_{t=1}^{T-1} [B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1})] + \frac{\eta}{2} \sum_{t=1}^T \|f'_t(x_t)\|_{\psi_t^*}^2. \quad (11)$$

The above corollaries allow us to prove regret bounds for a family of algorithms that iteratively modify the proximal functions ψ_t .

Algorithm 1 ADAGRAD with Diagonal Matrices

Input: $\eta > 0, \delta \geq 0$. Initialize $x_1 = 0, g_{1:0} = []$
for $t = 1$ to T **do**
 Suffer loss $f_t(x_t)$, receive subgradient $g_t \in \partial f_t(x_t)$ of f_t at x_t
 Update $g_{1:t} = [g_{1:t-1} \ g_t]$, $s_{t,i} = \|g_{1:t,i}\|_2$
 Set $H_t = \delta I + \text{diag}(s_t)$, $\psi_t(x) = \frac{1}{2} \langle x, H_t x \rangle$
 Primal-Dual Subgradient Update (Eq. (5)):

$$x_{t+1} = \operatorname{argmin}_{x \in \mathcal{X}} \left\{ \eta \left\langle \frac{1}{t} \sum_{\tau=1}^t g_\tau, x \right\rangle + \eta \varphi(x) + \frac{1}{t} \psi_t(x) \right\}.$$

Composite Mirror Descent Update (Eq. (6)):

$$x_{t+1} = \operatorname{argmin}_{x \in \mathcal{X}} \{ \eta \langle g_t, x \rangle + \eta \varphi(x) + B_{\psi_t}(x, x_t) \}.$$

end for

3 Diagonal Matrix Proximal Functions

For now we restrict ourselves to using diagonal matrices to define matrix proximal functions and (semi)norms. This restriction serves a two-fold purpose. First, the analysis for the general case is somewhat complicated and thus the analysis of the diagonal case serves as a proxy for better understanding. Second, in problems with high dimension where we expect this type of modification to help, maintaining more complicated proximal functions is likely to be prohibitively expensive. A benefit of the adaptive algorithms is that there is no need to keep track of a learning rate as in previous algorithms, as it is implicitly given by the growth of the proximal function. To remind the reader, $g_{1:t,i}$ is the i th row of the matrix obtained by concatenating the subgradients from iteration 1 through t in the online algorithm.

To provide some intuition for Alg. 1, let us find the retrospectively optimal proximal function. If the proximal function chosen is $\psi(x) = \frac{1}{2} \langle x, \text{diag}(s)x \rangle$ for some $s \succeq 0$, then the associated norm is $\|x\|^2 = \langle x, \text{diag}(s)x \rangle$ and the dual norm is $\|x\|_*^2 = \langle x, \text{diag}(s)^{-1}x \rangle$. Recalling Eq. (9), we consider the problem

$$\min_s \sum_{t=1}^T \langle g_t, \text{diag}(s)^{-1}g_t \rangle \quad \text{s.t. } s \succeq 0, \langle \mathbb{1}, s \rangle \leq c.$$

This problem is solved by setting $s_i = \|g_{1:T,i}\|_2$ and scaling s so that $\langle s, \mathbb{1} \rangle = c$. To see this, we can write the Lagrangian of the minimization problem by introducing multipliers $\lambda \succeq 0$ and $\theta \geq 0$ to get

$$\mathcal{L}(s, \lambda, \theta) = \sum_{i=1}^d \frac{\|g_{1:T,i}\|_2^2}{s_i} - \langle \lambda, s \rangle + \theta(\langle \mathbb{1}, s \rangle - c).$$

Taking derivatives to find the infimum of \mathcal{L} , we see that $-\|g_{1:T,i}\|_2^2/s_i^2 - \lambda_i + \theta = 0$, and the complementarity conditions (Boyd and Vandenberghe, 2004) on $\lambda_i s_i$ imply that $\lambda_i = 0$. Thus we have $s_i = \theta^{-\frac{1}{2}} \|g_{1:T,i}\|_2$, and normalizing using θ gives $s_i = c \|g_{1:T,i}\|_2 / \sum_{j=1}^d \|g_{1:T,j}\|_2$. As a final note, plugging s_i in gives

$$\inf_s \left\{ \sum_{t=1}^T \sum_{i=1}^d \frac{g_{t,i}^2}{s_i} : s \succeq 0, \langle \mathbb{1}, s \rangle \leq c \right\} = \frac{1}{c} \left(\sum_{i=1}^d \|g_{1:T,i}\|_2 \right)^2. \quad (12)$$

It is natural to suspect that if we use a proximal function similar to $\psi(x) = \frac{1}{2} \langle x, \text{diag}(s)x \rangle$, we should do well lowering the gradient terms in the regret in Eq. (10) and Eq. (11).

To prove a regret bound for our Alg. 1, we note that both types of updates have regret bounds including a term dependent solely on the gradients obtained along the algorithm's run. Thus, the following lemma, which says that the choice of ψ_t in Alg. 1 is optimal up to a multiplicative factor of 2, is applicable to both.

Lemma 4 Let $g_t = f'_t(x_t)$ and $g_{1:t}$ and s_t be defined as in Alg. 1. Then

$$\sum_{t=1}^T \langle g_t, \text{diag}(s_t)^{-1}g_t \rangle \leq 2 \sum_{i=1}^d \|g_{1:T,i}\|_2.$$

Proof: We prove the lemma by considering an arbitrary \mathbb{R} -valued sequence $\{a_i\}$ and its vector representation $a_{1:i} = [a_1 \ \cdots \ a_i]$. We are going to show that (where we set $0/0 = 0$)

$$\sum_{t=1}^T \frac{a_t^2}{\|a_{1:t}\|_2} \leq 2 \|a_{1:T}\|_2. \quad (13)$$

We use induction on T . For $T = 1$, the inequality trivially holds. Assume Eq. (13) holds true for $T - 1$, then

$$\sum_{t=1}^T \frac{a_t^2}{\|a_{1:t}\|_2} = \sum_{t=1}^{T-1} \frac{a_t^2}{\|a_{1:t}\|_2} + \frac{a_T^2}{\|a_{1:T}\|_2} \leq 2 \|a_{1:T-1}\|_2 + \frac{a_T^2}{\|a_{1:T}\|_2},$$

where the inequality follows from the inductive hypothesis. We now define $b_T = \sum_{t=1}^T a_t^2$ and use first-order inequality for concavity to obtain that so long as $b_T - a_T^2 \geq 0$, we have $\sqrt{b_T - a_T^2} \leq \sqrt{b_T} - a_T \frac{1}{2\sqrt{b_T}}$ (we use an identical technique in the full-matrix case; see Lemma 10). Thus

$$2 \|a_{1:T-1}\|_2 + \frac{a_T^2}{\|a_{1:T}\|_2} = 2\sqrt{b_T - a_T^2} + \frac{a_T^2}{\sqrt{b_T}} \leq 2\sqrt{b_T} = 2 \|a_{1:T}\|_2.$$

Having proved Eq. (13), we note that by construction $s_{t,i} = \|g_{1:t,i}\|_2$, thus,

$$\sum_{t=1}^T \langle g_t, \text{diag}(s_t)^{-1} g_t \rangle = \sum_{t=1}^T \sum_{i=1}^d \frac{g_{t,i}^2}{\|g_{1:t,i}\|_2} \leq 2 \sum_{i=1}^d \|g_{1:T,i}\|_2. \quad \blacksquare$$

To get a regret bound, we consider the terms consisting of the dual-norm of the subgradients in Eq. (10) and Eq. (11). When $\psi_t(x) = \langle x, (\delta I + \text{diag}(s_t))x \rangle$, the associated dual-norm is $\|g\|_{\psi_t^*}^2 = \langle g, (\delta I + \text{diag}(s_t))^{-1} g \rangle$. From the definition of s_t in Alg. 1, we clearly have $\|f'_t(x_t)\|_{\psi_t^*}^2 \leq \langle g_t, \text{diag}(s_t)^{-1} g_t \rangle$. We replace the inverse with a pseudo-inverse if needed, which is well defined since g_t is always in the column-space of $\text{diag}(s_t)$. Thus, Lemma 4 gives

$$\sum_{t=1}^T \|f'_t(x_t)\|_{\psi_t^*}^2 \leq 2 \sum_{i=1}^d \|g_{1:T,i}\|_2.$$

To obtain a bound for a primal-dual subgradient method, we set $\delta \geq \max_t \|g_t\|_\infty$, in which case $\|g_t\|_{\psi_{t-1}^*}^2 \leq \langle g_t, \text{diag}(s_t)^{-1} g_t \rangle$, and follow the same lines of reasoning.

It remains to bound the various Bregman divergence terms in Corollary 3 and the term $\psi_T(x^*)$ in Corollary 2. We focus first on composite mirror-descent updates. Examining Eq. (11) and Alg. 1, we notice that

$$\begin{aligned} B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1}) &= \frac{1}{2} \langle x^* - x_{t+1}, \text{diag}(s_{t+1} - s_t)(x^* - x_{t+1}) \rangle \\ &\leq \frac{1}{2} \max_i (x_i^* - x_{t+1,i})^2 \|s_{t+1} - s_t\|_1. \end{aligned}$$

Since $\|s_{t+1} - s_t\|_1 = \langle s_{t+1} - s_t, \mathbb{1} \rangle$ and $\langle s_T, \mathbb{1} \rangle = \sum_{i=1}^d \|g_{1:T,i}\|_2$, we have

$$\begin{aligned} \sum_{t=1}^{T-1} B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1}) &\leq \frac{1}{2} \sum_{t=1}^{T-1} \|x^* - x_{t+1}\|_\infty^2 \langle s_{t+1} - s_t, \mathbb{1} \rangle \\ &\leq \frac{1}{2} \max_{t \leq T} \|x^* - x_t\|_\infty^2 \sum_{i=1}^d \|g_{1:T,i}\|_2 - \frac{1}{2} \|x^* - x_1\|_\infty^2 \langle s_1, \mathbb{1} \rangle. \end{aligned} \quad (14)$$

We also have

$$\psi_T(x^*) = \delta \|x^*\|_2^2 + \langle x^*, \text{diag}(s_T)x^* \rangle \leq \delta \|x^*\|_2^2 + \|x^*\|_\infty^2 \sum_{i=1}^d \|g_{1:T,i}\|_2.$$

Combining the above arguments with Corollaries 2 and 3, and combining Eq. (14) with the fact that $B_{\psi_1}(x^*, x_1) \leq \frac{1}{2} \|x^* - x_1\|_\infty^2 \langle \mathbb{1}, s_1 \rangle$, we have proved the following theorem.

Theorem 5 Let the sequence $\{x_t\}$ be defined by Algorithm 1. If we generate x_t using the primal-dual subgradient update of Eq. (5) and $\delta \geq \max_t \|g_t\|_\infty$, then for any $x^* \in \mathcal{X}$ we have

$$R_\phi(T) \leq \frac{\delta}{\eta} \|x^*\|_2^2 + \frac{1}{\eta} \|x^*\|_\infty^2 \sum_{i=1}^d \|g_{1:T,i}\|_2 + \eta \sum_{i=1}^d \|g_{1:T,i}\|_2. \quad (15)$$

If we use Algorithm 1 with the composite mirror-descent update of Eq. (6), then for any $x^* \in \mathcal{X}$

$$R_\phi(T) \leq \frac{1}{2\eta} \max_{t \leq T} \|x^* - x_t\|_\infty^2 \sum_{i=1}^d \|g_{1:T,i}\|_2 + \eta \sum_{i=1}^d \|g_{1:T,i}\|_2. \quad (16)$$

The above theorem is a bit unwieldy. We thus perform a few algebraic simplifications to get the next corollary. Let us assume that \mathcal{X} is compact and set $D_\infty = \sup_{x \in \mathcal{X}} \|x - x^*\|_\infty$. Furthermore, define

$$\gamma_T = \sum_{i=1}^d \|g_{1:T,i}\|_2 = \sqrt{\inf_s \left\{ \sum_{t=1}^T \langle g_t, \text{diag}(s)^{-1} g_t \rangle : \langle 1, s \rangle \leq \sum_{i=1}^d \|g_{1:T,i}\|_2, s \succeq 0 \right\}}.$$

The following corollary is immediate.

Corollary 6 Assume that D_∞ and γ_T are defined as above. If we generate the sequence $\{x_t\}$ be given by Algorithm 1 using the primal-dual subgradient update Eq. (5) with $\eta = \|x^*\|_\infty$, then for any $x^* \in \mathcal{X}$

$$R_\phi(T) \leq 2 \|x^*\|_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2 + \delta \frac{\|x^*\|_2^2}{\|x^*\|_\infty} \leq 2 \|x^*\|_\infty \gamma_T + \delta \|x^*\|_1.$$

Using the composite mirror descent update of Eq. (6) to generate $\{x_t\}$ and setting $\eta = D_\infty/\sqrt{2}$, we have

$$R_\phi(T) \leq \sqrt{2} D_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2 = \sqrt{2} D_\infty \gamma_T.$$

We can also prove Corollary 1.

Proof of Corollary 1: The proof simply uses Theorem 5, Corollary 6, and the fact that

$$\inf_s \left\{ \sum_{t=1}^T \sum_{i=1}^d \frac{g_{t,i}^2}{s_i} : s \succeq 0, \langle 1, s \rangle \leq d \right\} = \frac{1}{d} \left(\sum_{i=1}^d \|g_{1:T,i}\|_2 \right)^2$$

as in Eq. (12) in the beginning of this section. Plugging the γ_T term in from Corollary 6 and multiplying D_∞ by \sqrt{d} completes the proof. \blacksquare

Intuitively, as discussed in the introduction, Alg. 1 should have good properties on sparse data. For example, suppose that our gradient terms are based on 0/1-valued features for a logistic regression task. Then the gradient terms in the bound $\sum_{i=1}^d \|g_{1:t,i}\|_2$ should all be much smaller than \sqrt{T} . If we assume that some features appear much more frequently than others, then the infimal representation of γ_T and the infimal equality in Corollary 1 show that we can have much lower learning rates on commonly appearing features and higher rates on uncommon features, and this will significantly lower the bound on the regret. Further, if we are constructing a relatively dense predictor x —as is often the case in sparse prediction problems—then $\|x^*\|_\infty$ is the best p -norm we can have in the regret.

4 Full Matrix Proximal Functions

In this section we derive and analyze new updates when we estimate a full matrix for the proximal function ψ_t instead of a diagonal one. In this generalized case, the algorithm uses the the square-root of the matrix of outer products of the gradients that observed to update the parameters. As in the diagonal case, we build on intuition garnered from an optimization problem. We seek a matrix S that solves the minimization problem

$$\min_S \sum_{t=1}^T \langle g_t, S^{-1} g_t \rangle \quad \text{s.t. } S \succeq 0, \quad \text{tr}(S) \leq c.$$

The solution is obtained by defining $G_t = \sum_{\tau=1}^t g_\tau g_\tau^\top$, and then setting S to be a normalized version of the root of G_T , that is, $S = c G_T^{1/2} / \text{tr}(G_T^{1/2})$. The next proposition formalizes this statement, and also shows that when G_T is not full rank we can instead use its pseudo-inverse. The proof is in Duchi et al. (2010a).

Algorithm 2 ADAGRAD with Full Matrices

Input $\eta > 0, \delta \geq 0$. Initialize $x = 0, S_0 = 0, H_0 = 0, G_0 = 0$

for $t = 1$ to T **do**

Suffer loss $f_t(x_t)$, receive subgradient $g_t \in \partial f_t(x_t)$ of f_t at x_t .

Update $G_t = G_{t-1} + g_t g_t^\top, S_t = G_t^{\frac{1}{2}}$.

Let $H_t = \delta I + S_t, \psi_t(x) = \frac{1}{2} \langle x, H_t x \rangle$

Primal-Dual Subgradient Update (Eq. (5))

$$x_{t+1} = \operatorname{argmin}_{x \in \mathcal{X}} \left\{ \eta \left\langle \frac{1}{t} \sum_{\tau=1}^t g_\tau, x \right\rangle + \eta \varphi(x) + \frac{1}{t} \psi_t(x) \right\}$$

Composite Mirror Descent Update (Eq. (6))

$$x_{t+1} = \operatorname{argmin}_{x \in \mathcal{X}} \{ \eta \langle g_t, x \rangle + \eta \varphi(x) + B_{\psi_t}(x, x_t) \}$$

end for

Proposition 7 Consider the following minimization problem:

$$\min_S \operatorname{tr}(S^{-1}A) \text{ subject to } S \succeq 0, \operatorname{tr}(S) \leq c \text{ where } A \succeq 0. \quad (17)$$

If A is of full rank, then the minimizer of Eq. (17) is $S = cA^{\frac{1}{2}} / \operatorname{tr}(A^{\frac{1}{2}})$. If A is not of full rank, then setting $S = cA^{\frac{1}{2}} / \operatorname{tr}(A^{\frac{1}{2}})$ gives

$$\operatorname{tr}(S^\dagger A) = \inf_S \{ \operatorname{tr}(S^{-1}A) : S \succeq 0, \operatorname{tr}(S) \leq c \}.$$

In either case, $\operatorname{tr}(S^\dagger A) = \operatorname{tr}(A^{\frac{1}{2}})^2 / c$.

If we iteratively use proximal functions of the form $\psi_t(x) = \langle x, G_t^{1/2} x \rangle$, we hope as earlier to attain low regret and collect gradient information. We achieve our low regret goal by employing a similar doubling lemma to Lemma 4. The resulting algorithm is given in Alg. 2, and the next theorem provides a quantitative analysis of the motivation above.

Theorem 8 Let G_t be the outer product matrix defined above. If we generate x_t using the primal-dual subgradient update of Eq. (5) and $\delta \geq \max_t \|g_t\|_2$, then for any $x^* \in \mathcal{X}$

$$R_\phi(T) \leq \frac{\delta}{\eta} \|x^*\|_2^2 + \frac{1}{\eta} \|x^*\|_2^2 \operatorname{tr}(G_T^{1/2}) + \eta \operatorname{tr}(G_T^{1/2}). \quad (18)$$

If we use Algorithm 2 with the composite mirror-descent update of Eq. (6), then for any x^* and $\delta \geq 0$

$$R_\phi(T) \leq \frac{\delta}{\eta} \|x^*\|_2^2 + \frac{1}{2\eta} \max_{t \leq T} \|x^* - x_t\|_2^2 \operatorname{tr}(G_T^{1/2}) + \eta \operatorname{tr}(G_T^{1/2}). \quad (19)$$

Proof: To begin, we consider the difference between the divergence terms at time $t + 1$ and time t from Eq. (11) in Corollary 3. Let $\lambda_{\max}(M)$ denote the largest eigenvalue of a matrix M . We have

$$\begin{aligned} B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1}) &= \frac{1}{2} \left\langle x^* - x_{t+1}, (G_{t+1}^{1/2} - G_t^{1/2})(x^* - x_{t+1}) \right\rangle \\ &\leq \frac{1}{2} \|x^* - x_{t+1}\|_2^2 \lambda_{\max}(G_{t+1}^{1/2} - G_t^{1/2}) \leq \frac{1}{2} \|x^* - x_{t+1}\|_2^2 \operatorname{tr}(G_{t+1}^{1/2} - G_t^{1/2}). \end{aligned}$$

For the last inequality we used the fact that the trace of a matrix is equal to the sum of its eigenvalues along with the property $G_{t+1}^{1/2} - G_t^{1/2} \succeq 0$ (Davis, 1963, Example 3) and therefore $\operatorname{tr}(G_{t+1}^{1/2} - G_t^{1/2}) \geq \lambda_{\max}(G_{t+1}^{1/2} - G_t^{1/2})$. Thus, we get

$$\begin{aligned} \sum_{t=1}^{T-1} B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1}) &\leq \frac{1}{2} \sum_{t=1}^{T-1} \|x^* - x_{t+1}\|_2^2 \left(\operatorname{tr}(G_{t+1}^{1/2}) - \operatorname{tr}(G_t^{1/2}) \right) \\ &\leq \frac{1}{2} \max_{t \leq T} \|x^* - x_t\|_2^2 \operatorname{tr}(G_T^{1/2}) - \frac{1}{2} \|x^* - x_1\|_2^2 \operatorname{tr}(G_1^{1/2}). \quad (20) \end{aligned}$$

For the last inequality we used the fact that G_1 is a rank 1 PSD matrix with non-negative trace. What remains is to bound the gradient terms common to both updates. The following lemma is directly applicable.

Lemma 9 Let $S_t = G_t^{1/2}$ be as defined in Alg. 2. Then, using the pseudo-inverse when necessary,

$$\sum_{t=1}^T \langle g_t, S_t^{-1} g_t \rangle \leq 2 \sum_{t=1}^T \langle g_t, S_T^{-1} g_t \rangle = 2 \operatorname{tr}(G_T^{1/2}).$$

Before we prove the lemma, we state two linear-algebraic lemmas that make its proof and that of the theorem much more straightforward. The lemmas are quite technical, so we prove them in the long version of this paper (Duchi et al., 2010a). The first auxiliary lemma is the matrix-analogue of the fact that for nonnegative x, y with $x \geq y$, $\sqrt{x-y} \leq \sqrt{x-y}/(2\sqrt{x})$, a consequence of the concavity of $\sqrt{\cdot}$.

Lemma 10 Let $B \succeq 0$ and $B^{-1/2}$ denote the root of the inverse (or pseudo-inverse) of B . For any c such that $B - cgg^\top \succeq 0$, the following inequality holds:

$$2 \operatorname{tr}((B - cgg^\top)^{1/2}) \leq 2 \operatorname{tr}(B^{1/2}) - c \operatorname{tr}(B^{-1/2} gg^\top).$$

Lemma 11 Let $\delta \geq \|g\|_2$ and $A \succeq 0$. Then $\langle g, (\delta I + A^{1/2})^{-1} g \rangle \leq \langle g, ((A + gg^\top)^\dagger)^{1/2} g \rangle$.

Proof of Lemma 9: We prove the lemma by induction. The base case is immediate, since $\langle g_1, G_1^{-1/2} g_1 \rangle = \frac{\langle g_1, g_1 \rangle}{\|g_1\|_2} = \|g_1\|_2 \leq 2 \|g_1\|_2$. Now, assume the lemma is true for $T-1$, so from the inductive assumption

$$\sum_{t=1}^T \langle g_t, S_t^{-1} g_t \rangle \leq 2 \sum_{t=1}^{T-1} \langle g_t, S_{T-1}^{-1} g_t \rangle + \langle g_T, S_T^{-1} g_T \rangle.$$

Since S_{T-1} does not depend on t , $\sum_{t=1}^{T-1} \langle g_t, S_{T-1}^{-1} g_t \rangle = \operatorname{tr}(S_{T-1}^{-1} \sum_{t=1}^{T-1} g_t g_t^\top) = \operatorname{tr}(G_{T-1}^{-1/2} G_{T-1})$, where the right-most equality follows from the definitions of S_t and G_t . Therefore, we get

$$\sum_{t=1}^T \langle g_t, S_t^{-1} g_t \rangle \leq 2 \operatorname{tr}(G_{T-1}^{-1/2} G_{T-1}) + \langle g_T, G_T^{-1/2} g_T \rangle = 2 \operatorname{tr}(G_T^{1/2}) + \langle g_T, G_T^{-1/2} g_T \rangle.$$

Lemma 10, which also justifies the use of pseudo-inverses, lets us exploit the concavity of the function $\operatorname{tr}(A^{1/2})$ to bound the above sum by $2 \operatorname{tr}(G_T^{1/2})$. \blacktriangle

We can now finalize our proof of the theorem. As in the diagonal case, we have that the squared dual norm (seminorm when $\delta = 0$) associated with ψ_t is

$$\|x\|_{\psi_t^*}^2 = \langle x, (\delta I + S_t)^{-1} x \rangle.$$

Thus it is clear that $\|g_t\|_{\psi_t^*}^2 \leq \langle g_t, S_t^{-1} g_t \rangle$. For the dual-averaging algorithms, we use Lemma 11 to see that $\|g_t\|_{\psi_{t-1}^*}^2 \leq \langle g_t, S_t^{-1} g_t \rangle$ so long as $\delta \geq \|g_t\|_2$. The doubling inequality from Lemma 9 implies that $\sum_{t=1}^T \|f'_t(x_t)\|_{\psi_t^*}^2 \leq 2 \operatorname{tr}(G_T^{1/2})$ for mirror-descent algorithms and that $\sum_{t=1}^T \|f'_t(x_t)\|_{\psi_{t-1}^*}^2 \leq 2 \operatorname{tr}(G_T^{1/2})$ for primal-dual subgradient algorithms.

Note that $B_{\psi_1}(x^*, x_1) \leq \frac{1}{2} \|x^* - x_1\|_2^2 \operatorname{tr}(G_1^{1/2})$ when $\delta = 0$. Combining the first of the last bounds in the previous paragraph with this and the bound on $\sum_{t=1}^{T-1} B_{\psi_{t+1}}(x^*, x^{t+1}) - B_{\psi_t}(x^*, x^{t+1})$ from Eq. (20), we see that Corollary 3 gives the bound for the mirror-descent family of algorithms. Combining the second of the bounds in the previous paragraph and Eq. (20) with Corollary 2 gives the desired bound on $R_\phi(T)$ for the primal-dual subgradient algorithms, which completes the proof of the theorem. \blacksquare

As before, we give a corollary that clarifies the bound implied by Theorem 8. The infimal equalities in the corollary use Proposition 7. The corollary suggests that if there is a rotation of the space in which the gradient vectors g_t have small inner products—a sparse basis for the subgradients g_t —then using full-matrix proximal functions can significantly lower the regret.

Corollary 12 The sequence $\{x_t\}$ generated by Alg. 2 with the primal-dual update and $\eta = \|x^*\|_2$ satisfies

$$R_\phi(T) \leq 2 \|x^*\|_2 \operatorname{tr}(G_T^{1/2}) + \delta \|x^*\|_2 = 2\sqrt{d} \|x^*\|_2 \sqrt{\inf_S \left\{ \sum_{t=1}^T \langle g_t, S^{-1} g_t \rangle : S \succeq 0, \operatorname{tr}(S) \leq d \right\}} + \delta \|x^*\|_2.$$

Let \mathcal{X} be compact so that $\sup_{x \in \mathcal{X}} \|x - x^*\|_2 \leq D_2$. Let $\eta = D_2/\sqrt{2}$ and $\{x_t\}$ be generated by Alg. 2 using the composite mirror descent update with $\delta = 0$. Then

$$R_\phi(T) \leq \sqrt{2} D_2 \operatorname{tr}(G_T^{1/2}) = \sqrt{2} d D_2 \sqrt{\inf_S \left\{ \sum_{t=1}^T \langle g_t, S^{-1} g_t \rangle : S \succeq 0, \operatorname{tr}(S) \leq d \right\}}.$$

5 Lowering the Regret for Strongly Convex Functions

It is now well established that strong convexity of the functions f_t can give significant improvements in the regret of online convex optimization algorithms (Hazan et al., 2006; Shalev-Shwartz and Singer, 2007). We can likewise derive lower regret bounds in the presence of strong convexity. We assume that our functions $f_t + \varphi$ are strongly convex with respect to a norm $\|\cdot\|$. For simplicity, we assume that each has the same strong convexity parameter λ ,

$$f_t(y) + \varphi(y) \geq f_t(x) + \varphi(x) + \langle f'_t(x), y - x \rangle + \langle \varphi'(x), y - x \rangle + \frac{\lambda}{2} \|x - y\|^2.$$

We focus on composite mirror descent algorithms, as the analysis of strongly convex variants of primal-dual subgradient algorithms does not seem to lend itself to dynamic learning rate adaptation. The tightest analysis of the primal-dual method for strongly-convex functions keeps the function ψ intact rather than growing it at a rate of \sqrt{t} , as in standard RDA (Xiao, 2009). Allowing ψ to grow makes attaining the stronger regret bound impossible. It may be possible to analyze RDA when the regularization function φ is time-dependent, but we leave this topic to future research. Without loss of generality let $\varphi(x_1) = 0$ and $x_1 = 0$. Rather than give the proof of the lower regret, we simply state the result, as it is not difficult to prove using techniques of Hazan et al. (2006), though we include the proof in the full version of this paper (Duchi et al., 2010a).

Theorem 13 *Assume that φ is λ -strongly convex with respect to $\|\cdot\|_2^2$ over the set \mathcal{X} . Assume further that $\|g\|_\infty \leq G_\infty$ for all $g \in \partial f_t(x)$ for $x \in \mathcal{X}$. Let $\{x_t\}$ be the sequence of vectors generated by Algorithm 1 with the diagonal proximal function $\psi_t(x) = \langle x, (\delta I + \text{diag}(s_t))x \rangle$ and $s_{t,i} = \|g_{1:t,i}\|_2^2$. Setting $\eta \geq \frac{G_\infty^2}{\lambda}$, the regret is bounded by*

$$R_\phi(T) \leq \frac{2G_\infty^2 \delta}{\lambda} \|x_1 - x^*\|_2^2 + \frac{G_\infty^2}{\lambda} \sum_{i=1}^d \log \left(\frac{\|g_{1:T,i}\|_2^2}{\delta} + 1 \right) = O \left(\frac{dG_\infty^2}{\lambda} \log(TG_\infty) \right).$$

6 Experiments

In this section, we present the results of experiments with natural datasets that suggest that adaptive methods significantly outperform related non-adaptive methods. We focus on the fully stochastic optimization setting, in which at each iteration the learning algorithm receives a single example. We measure performance using two metrics: the online loss or error and the test set performance of the predictor the learning algorithm outputs at the end of a single pass through the training data. We also give some results that show how imposing sparsity constraints (in the form of ℓ_1 and mixed-norm regularization) affects the learning algorithm's performance. One benefit of the ADAGRAD framework is its ability to straightforwardly generalize to domain constraints $\mathcal{X} \neq \mathbb{R}^d$ and arbitrary regularization functions φ , in contrast to previous adaptive online algorithms. See Duchi et al. (2010a) for a more complete experimental evaluation.

We experiment with RDA (Xiao, 2009), FOBOS (Duchi and Singer, 2009), adaptive RDA, adaptive FOBOS, the Passive-Aggressive (PA) algorithm (Crammer et al., 2006), and AROW (Crammer et al., 2009). To remind the reader, PA is an online learning procedure with the update

$$x_{t+1} = \underset{x}{\operatorname{argmin}} [1 - y_t \langle z_t, x \rangle]_+ + \frac{\lambda}{2} \|x - x_t\|_2^2,$$

where λ is a regularization parameter. PA's update is similar to the update employed by AROW (see Eq. (8)), but the latter maintains second order information on x . Using the representer theorem, it is also possible to derive efficient updates for PA and AROW for the logistic loss, $\log(1 + \exp(-y_t \langle z_t, x_t \rangle))$. We thus compare the above six algorithms using both hinge and logistic loss.

The Reuters RCV1 dataset is a collection of approximately 800,000 text articles, each of which is assigned multiple labels. There are 4 high-level categories—Economics, Commerce, Medical, and Government (ECAT, CCAT, MCAT, GCAT)—and multiple more specific categories. We focus on training binary classifiers for each of the four major categories. The input features we use are 0/1 bigram features, which (post word stemming) yield a representation of approximately 2 million dimensions. The feature vectors are very sparse, however, and most examples have fewer than 5000 non-zero features.

We compare the twelve different algorithms mentioned in the prequel as well as variants of FOBOS and RDA with ℓ_1 -regularization. We summarize the results of the ℓ_1 -regularized runs as well as AROW and PA in Table 1. We found the results for both the hinge loss and the logistic loss to be qualitatively and quantitatively very similar. We thus report results only for training with the hinge loss in Table 1. Each row in the table represents the average of four different experiments in which we hold out 25% of the data for test and perform a single online learning pass on the remaining 75% of the data. For RDA and FOBOS, we cross-validate the

	RDA	FB	ADAGRAD-RDA	ADAGRAD-FB	PA	AROW
ECAT	.051 (.099)	.058 (.194)	.044 (.086)	.044 (.238)	.059	.049
CCAT	.064 (.123)	.111 (.226)	.053 (.105)	.053 (.276)	.107	.061
GCAT	.046 (.092)	.056 (.183)	.040 (.080)	.040 (.225)	.066	.044
MCAT	.037 (.074)	.056 (.146)	.035 (.063)	.034 (.176)	.053	.039

Table 1: Test set error rates and proportion non-zero weights (in parenthesis) on Reuters RCV1.

stepsize parameter η by running multiple passes and then choosing the output of the learner that had the fewest mistakes during training. For PA and AROW we choose λ using the same approach. We use the same regularization multiplier for the ℓ_1 term to execute RDA and FOBOS. The regularization multiplier was selected so that RDA yielded a weight vector with approximately 10% non-zero components.

It is evident from the results presented in Table 1 that the adaptive algorithms (AROW and ADAGRAD) are far superior to non-adaptive algorithms in terms of error rate on test data. In addition, the ADAGRAD algorithms naturally incorporate sparsity since they were run with ℓ_1 -regularization, though RDA obtained significantly higher sparsity levels while the solutions of PA and AROW are dense. Furthermore, although omitted from the table for brevity, in *every* test with the RCV1 corpus, the adaptive algorithms outperformed the non-adaptive algorithms. Moreover, both ADAGRAD-RDA and ADAGRAD-Fobos outperform AROW on all the classification tasks. Unregularized RDA and FOBOS attained similar results to the ℓ_1 -regularized variants, though of course the solution of the former versions were not sparse.

7 Conclusions

We presented a paradigm that adapts subgradient methods to the geometry of the problem at hand. The adaptation allows us to derive strong regret guarantees, which for some natural data distributions achieve better performance guarantees than previous algorithms. Our online convergence results can be naturally converted into rate of convergence and generalization bounds (Cesa-Bianchi et al., 2004). The ADAGRAD family of algorithms incorporates regularization through φ and can thus easily generate sparse or otherwise structured solutions. Our algorithms are straightforward to implement and can be easily specialized to many useful constraint sets \mathcal{X} and regularization terms φ . We conducted comprehensive experiments showing that adaptive methods clearly outperform their non-adaptive counterparts. These results are available in the long version of this paper (Duchi et al., 2010a). We believe that there are a few theoretical questions that are still unanswered in this line of work. The first is whether we can *efficiently* use full matrices in the proximal functions, as in Section 4, or whether a different algorithm is necessary. A second open issue is whether it is possible to use non-Euclidean proximal functions. For example, is it possible to adapt the KL divergence between distributions to characteristics of the problem at hand? We hope to investigate such extensions in the near future.

Acknowledgments

The first author was supported by the Department of Defense (DoD) through the National Defense Science and Engineering Graduate Fellowship program. Part of this work was carried out while the first author was working at Google Research.

References

- J. Abernethy, P. Bartlett, A. Rakhlin, and A. Tewari. Optimal strategies and minimax lower bounds for online convex games. In *Proceedings of the Twenty First Annual Conference on Computational Learning Theory*, 2008.
- P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 64(1):48–75, 2002.
- P. L. Bartlett, E. Hazan, and A. Rakhlin. Adaptive online gradient descent. In *Advances in Neural Information Processing Systems 20*, 2007.
- A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 2003.
- A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.
- S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

- N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, September 2004.
- N. Cesa-Bianchi, A. Conconi, , and C. Gentile. A second-order perceptron algorithm. *SIAM Journal on Computing*, 34(3):640–668, 2005.
- K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.
- K. Crammer, M. Dredze, and A. Kulesza. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems 23*, 2009.
- C. Davis. Notions generalizing convexity for functions defined on spaces of matrices. In *Proceedings of the Symposia in Pure Mathematics*, volume 7, pages 187–201. American Mathematical Society, 1963.
- J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2873–2908, 2009.
- J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report 2010-24, UC Berkeley Electrical Engineering and Computer Science, 2010a. URL cs.berkeley.edu/~jduchi/projects/DuchiHaSi10.pdf.
- J. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. Composite objective mirror descent. In *Proceedings of the Twenty Third Annual Conference on Computational Learning Theory*, 2010b.
- R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970.
- E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006.
- A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2003.
- Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1): 221–259, 2009.
- G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5), 1988.
- S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. Technical report, The Hebrew University, 2007. URL <http://www.cs.huji.ac.il/~shais>.
- N. Z. Shor. Utilization of the operation of space dilation in the minimization of convex functions. *Cybernetics and Systems Analysis*, 6(1):7–15, 1972. Translated from *Kibernetika*.
- P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. Technical report, Department of Mathematics, University of Washington, 2008.
- L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. In *Advances in Neural Information Processing Systems 23*, 2009.
- M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

Characterization of Linkage-based Clustering

Margareta Ackerman Shai Ben-David David Loker

D.R.C School of Computer Science

University of Waterloo

{mackerma, shai, dloker}@cs.uwaterloo.ca

Abstract

Clustering is a central unsupervised learning task with a wide variety of applications. Not surprisingly, there exist many clustering algorithms. However, unlike classification tasks, in clustering, different algorithms may yield dramatically different outputs for the same input sets. A major challenge is to develop tools that may help select the more suitable algorithm for a given clustering task. We propose to address this problem by distilling abstract properties of clustering functions that distinguish between the types of input-output behaviors of different clustering paradigms. In this paper we make a significant step in this direction by providing such property based characterization for the class of linkage based clustering algorithms.

Linkage-based clustering is one the most commonly used and widely studied clustering paradigms. It includes popular algorithms like Single Linkage and enjoys simple efficient algorithms.

On top of their potential merits for helping users decide when are such algorithms appropriate for their data, our results can be viewed as a convincing proof of concept for the research on taxonomizing clustering paradigms by their abstract properties.

1 Introduction

Having a clustering task at hand, a user needs to choose from a wide variety of clustering algorithms that, when run on the same input data, often produce very different clusterings. In spite of the wide use of clustering in many practical applications, the practices of choosing an algorithm for a given task are completely ad hoc – currently, there exists no principled method to guide the selection of a clustering algorithm. The choice of an appropriate clustering should, of course, be task dependent. A clustering that works well for one task may be unsuitable for another. Even more than for supervised learning, for clustering, the choice of an algorithm must incorporate domain knowledge. A major challenge that has hardly been addressed is how to turn domain knowledge into useful guidance to the clustering algorithm designer. One approach to providing guidance to clustering users in the selection of a clustering algorithm is to identify significant properties of clustering functions that, on one hand distinguish between different clustering paradigms, and on the other hand are relevant to the domain knowledge that a user might have. Based on domain expertise users could then choose which properties they want an algorithm to satisfy, and determine which algorithms satisfy each of these properties.

Ultimately, there would be a sufficiently rich set of properties that would provide detailed, property based, taxonomy of clustering methods, that could, in turn, be used as guidelines for a wide variety of clustering users.

One of the most basic challenges along this line is to find abstract properties of clustering functions that distinguish between the major families of common clustering algorithms, such as linkage based algorithms, center based algorithms and spectral clustering algorithms. Bosagh Zadeh and Ben-David [2] made progress in this direction by providing a set of abstract properties that characterize the single linkage clustering method.

In this paper we succeed in making the next major step by distilling a set of abstract properties that distinguish between linkage based clustering to any other type of clustering paradigm. Linkage-based clusterings is a family of clustering methods that include some of the most commonly-used and widely-studied clustering paradigms. We provide a surprisingly simple set of properties that, on one hand is satisfied by all the algorithm in that family, while on the other hand, no algorithm outside that family satisfies (all of) the properties in that set. Our characterization highlights the way in which the clusterings that are output by linkage-based algorithms are different from the clusterings output by other clustering algorithms.

On top of the importance of understanding linkage based clustering, we hope that this analysis will serve as a proof of concept to the research direction of distilling clustering properties towards providing useful guidelines for selecting clustering algorithms in practical applications.

2 Previous Work

Our work follows a theoretical study of clustering that began with Kleinberg's impossibility result [7], in which he proposes three axioms of clustering and shows that no clustering function can simultaneously satisfy these three axioms. Ackerman and Ben-David [1] subsequently showed these axioms to be consistent in the setting of clustering quality measures. Additionally, they discuss the distinction between axioms and properties, addressing the question of when a property is an axiom. In particular, while a property may be satisfied by some, but not all, objects of a class, an axiom is a property that is satisfied by all objects of the class. In the current paper we propose variations of some of Kleinberg's axioms and use these variations in our characterization of linkage-based clustering.

There are a couple of previous characterizations of the single-linkage algorithm. In 1975, Jardine and Sibson [6] gave a characterization of single linkage. More recently, Bosagh Zadeh and Ben-David [2] characterize single-linkage within Kleinberg's framework of clustering functions. To the best of our knowledge, ours is the first characterization of a commonly used class of clustering algorithms. We note that although single-linkage is a linkage-based algorithm, our characterization doesn't build upon the previous characterizations mentioned.

3 Preliminaries and Notation

Clustering is a very wide and heterogenous domain. We choose to focus on a basic sub-domain where the (only) input to the clustering function is a finite set of points endowed with a between-points distance (or similarity) function, and the output is a partition of that domain. This sub-domain is rich enough to capture many of the fundamental issues of clustering, while keeping the underlying structure as succinct as possible.

A *distance function* is a symmetric function $d : X \times X \rightarrow R^+$, such that $d(x, x) = 0$ for all $x \in X$.

The objects that we consider are pairs (X, d) , where X is some finite domain set and d is a distance function d over X . These are the inputs for clustering functions.

At times we consider a domain subset with the distance induced from the full domain set. We let $(X', d') \subseteq (X, d)$ denote $X' \subseteq X$ and $d' = d|_{X'}$, is defined by restricting the distance function d to $(X')^2$.

We say that a distance function d over X *extends* distance function d' over $X' \subseteq X$ if $d' \subseteq d$.

A *k-clustering* $C = \{c_1, c_2, \dots, c_k\}$ of data set X is a partition of X into k disjoint subsets of X (so, $\bigcup_i c_i = X$). A *clustering* of X is a k -clustering of X for some $1 \leq k \leq |X|$.

For a clustering C , let $|C|$ denote the number of clusters in C . For $x, y \in X$ and clustering C of X , we write $x \sim_C y$ if x and y belong to the same cluster in C and $x \not\sim_C y$, otherwise.

Definition 1 (Isomorphisms between domain sets) *Two notions of isomorphism of structures are relevant to our discussion.*

1. We say that (X, d) and (X', d') are *isomorphic domains*, denoting it by $(X, d) \sim (X', d')$, if there exists a bijection $\phi : X \rightarrow X'$ so that $d(x, y) = d'(\phi(x), \phi(y))$ for all $x, y \in X$.
2. We say that to clusterings (or partitions) $C = (c_1, \dots, c_k)$ of some domain (X, d) and $C' = (c'_1, \dots, c'_k)$ of some domain (X', d') are *isomorphic clusterings*, denoted $(C, d) \cong (C', d')$, if there exists a bijection $\phi : X \rightarrow X'$ such that for all $x, y \in X$, $d(x, y) = d'(\phi(x), \phi(y))$ and, on top of that, $x \sim_C y$ if and only if $\phi(x) \sim_{C'} \phi(y)$. Note that this notion depends on both the underlying distance functions and the clusterings.

Definition 2 (Clustering functions) *A clustering function is a function that takes as input a pair (X, d) and a parameter $1 \leq k \leq |X|$ and outputs a k -clustering of the domain X . We require such a function, F , to satisfy the following:*

1. **Representation Independence:** *Whenever $(X, d) \sim (X', d')$, then, for every k , $F(X, d, k)$ and $F(X', d', k)$ are isomorphic clusterings.*
2. **Scale Invariance:** *For any domain set X and any pair of distance functions d, d' over X , if there exists $c \in R^+$ such that $d(a, b) = c \cdot d'(a, b)$ for all $a, b \in X$, then $F(X, d, k) = F(X, d', k)$.*

4 Defining Linkage-Based Clustering

A linkage-based algorithm begins by placing every element of the input data set into its own cluster, and then repeatedly merging the “closest” clusters. What distinguishes different linkage-based algorithms from each other is the definition of between-cluster distance, which is used to determine the closest clusters. For example, *single linkage* defines cluster distance by the shortest edge between members of the clusters, while *complete linkage* uses the longest between cluster edge to define the distance between clusters.

Between-cluster distance has been formalized in a variety of ways. It has been called a “linkage function,” (see, for example, [3] and [5]). Everitte et al. [4] call it “inter-object distance.” Common to all these formalisms is function that maps pairs of clusters to real numbers. No further detailing of the concept has been previously explored. In this paper, we zoom in on the concept of between-cluster distance and provide a rigorous, general definition.

Definition 3 (Linkage function) A linkage function is a function

$$\ell : \{(X_1, X_2, d) \mid d \text{ is a distance function over } X_1 \cup X_2\} \rightarrow \mathcal{R}^+$$

such that,

1. ℓ is representation independent: For all (X_1, X_2) and (X'_1, X'_2) , if $(X_1, X_2, d) \cong (X'_1, X'_2, d')$ (i.e., they are clustering-isomorphic), then $\ell(X_1, X_2, d) = \ell(X'_1, X'_2, d')$.
2. ℓ is monotonic: For all (X_1, X_2, d) if d' is a distance function over $X_1 \cup X_2$ such that for all $x \sim_{\{X_1, X_2\}} y$, $d(x, y) = d'(x, y)$ and for all $x \not\sim_{\{X_1, X_2\}} y$, $d(x, y) \leq d'(x, y)$ then $\ell(X_1, X_2, d') \geq \ell(X_1, X_2, d)$.
3. Any pair of clusters can be made arbitrarily distant: For any pair of data sets (X_1, d_1) , (X_2, d_2) , and any r in the range of ℓ , there exists a distance function d that extends d_1 and d_2 such that $\ell(X_1, X_2, d) > r$.

For technical reasons, we shall assume that a linkage function has a countable range. Say, the set of non-negative algebraic real numbers¹.

Note that a linkage function is only given the data for two clusters, as such, the distance between two clusters does not depend on data that is outside these clusters. Condition (1) formalizes the requirement that the distance does not depend on the labels (or identities) of domain points. The between-cluster distance is fully determined by the matrix of between-points distances. Conditions (2) and (3) relate the linkage function to the input distance function, and capture the intuition that pulling the points of one cluster further apart from those of another cluster, would not make the two clusters closer. Property (4) captures the intuition that by pulling two clusters away from each other they can be made arbitrarily “unlinked”.

We now define linkage-based clustering functions.

Definition 4 (linkage-based clustering function) A clustering function F is linkage-based if there exists a linkage function ℓ so that

- $F(X, d, |X|) = \{\{x\} \mid x \in X\}$
- For $1 \leq k < |X|$, $F(X, d, k)$ is constructed by merging the two clusters in $F(X, d, k+1)$ that minimize the value of ℓ . Formally,

$$F(X, d, k) = \{c \mid c \in F(X, d, k+1), c \neq c_i, c \neq c_j\} \cup \{c_i \cup c_j\},$$

such that $\{c_i, c_j\} = \operatorname{argmin}_{\{c_i, c_j\} \subseteq F(X, d, k+1)} \ell(c_i, c_j, d)$.

Here are examples of linkage functions used in the most common linkage-based algorithms.

- *Single linkage*: $\ell_{SL}(A, B, d) = \min_{a \in A, b \in B} d(a, b)$.
- *Average linkage*: $\ell_{AL}(A, B, d) = \frac{\sum_{a \in A, b \in B} d(a, b)}{|A| \cdot |B|}$
- *Complete linkage*: $\ell_{CL}(A, B, d) = \max_{a \in A, b \in B} d(a, b)$.

Note that ℓ_{SL} , ℓ_{AL} , and ℓ_{CL} satisfy the conditions of Definition 3 and as such are linkage functions².

¹Imposing this restriction simplifies our main proof, while not having any meaningful impact on the scope of clusterings considered

²A tie breaking mechanism is often used to apply such linkage functions in practice. For simplicity, we assume in this discussion that no ties occur. In other words, we assume that the linkage function is one-to-one on the set of isomorphism-equivalence classes of pairs of clusters.

5 Properties of Clustering Functions

We now introduce properties of clustering functions that we use to characterize linkage-based clustering.

5.1 Hierarchical clustering

The term Hierarchical clustering is a widely used to denote clustering algorithms that operate in a "bottom up" manner, starting from singleton clusters and creating coarser and coarser clusterings by merging clusters. Sometimes, it is also used to denote the more specific family of linkage-based clustering algorithms. Here we offer a precise formalization on what makes a clustering algorithm hierarchical.

Definition 5 (Clustering Refinement) A clustering C of X is a refinement of clustering C' of X if every cluster in C is a subset of some cluster in C' , or, equivalently, if every cluster of C' is a union of clusters of C .

Definition 6 (Hierarchical Functions) A clustering function is hierarchical if for every $1 \leq k \leq k' \leq |X|$, $F(X, d, k')$ is a refinement of $F(X, d, k)$.

5.2 Locality

We now introduce a new property of clustering algorithms that we call "locality". Intuitively, a clustering function is local if its behavior on a union of a subset of the clusters (in a clustering it outputs) depends only on distances between elements of that union, and is independent of the rest of the domain set.

Definition 7 (Locality) A clustering function F is local if for any clustering C output by F and every subset of clusters, $C' \subseteq C$,

$$F\left(\bigcup C', d, |C'|\right) = C'.$$

In other words, for every domain (X, d) and number of clusters, k , if X' is the union of k' clusters in $F(X, d, k)$ for some $k' \leq k$, then, applying F to (X', d) and asking for a k' -clustering, will yield the same clusters that we started with.

To better understand locality, consider two runs of a clustering algorithm. In the first run, the algorithm is called on some data set X and returns a k -clustering C . We then select some clusters $c_1, c_2, \dots, c_{k'}$ of C , and run the clustering algorithm on the points that the selected clusters consist of, namely, $c_1 \cup c_2 \cup \dots \cup c_{k'}$ asking for k' clusters. If the algorithm is local, then on the second run of the algorithm it will output $\{c_1, c_2, \dots, c_{k'}\}$.

5.3 Consistency

Consistency, introduced by Kleinberg [7], requires that the output of a clustering function, be invariant to shrinking within-cluster distances, and stretching between-cluster distances.

Definition 8 (consistency) Given a clustering C of some domain (X, d) , we say that a distance function d' over X , is C, d -consistent if

1. $d'_X(x, y) \leq d_X(x, y)$ whenever $x \sim_C y$, and
2. $d'_X(x, y) \geq d_X(x, y)$ whenever $x \not\sim_C y$.

A clustering function F is consistent if for every X, d, k , if d' is $(F(X, d, k), d)$ -consistent then $F(X, dk) = F(X, d', k)$.

We introduce two relaxations of consistency.

Definition 9 (outer-consistency) Given a clustering C of some domain (X, d) , we say that a distance function d' over X , is (C, d) -outer consistent if

1. $d'_X(x, y) = d_X(x, y)$ whenever $x \sim_C y$, and
2. $d'_X(x, y) \geq d_X(x, y)$ whenever $x \not\sim_C y$.

A clustering function F is outer consistent if for every X, d, k , if d' is $(F(X, d, k), d)$ -outer consistent then $F(X, d, k) = F(X, d', k)$.

Definition 10 (inner-consistency)

Given a clustering C of some domain (X, d) , we say that a distance function d' over X , is (C, d) -inner consistent if

1. $d'_X(x, y) \leq d_X(x, y)$ whenever $x \sim_C y$, and
2. $d'_X(x, y) = d_X(x, y)$ whenever $x \not\sim_C y$.

A clustering function F is inner consistent if for every X, d, k , if d' is $(F(X, d, k), d)$ - inner consistent then $F(X, d, k) = F(X, d', k)$.

Clearly, consistency implies both outer-consistency and inner-consistency.

Outer-consistency is satisfied by many common clustering functions. In Lemma 26, we will show that any linkage-based clustering function is outer-consistent. We also show below that the average-linkage and complete linkage clustering functions are not inner consistent, and therefore (since they satisfy the other two of Kleinberg's axioms, and no function satisfies all three axioms) they are not consistent.

5.4 Richness

Kleinberg introduced a Richness property as one of his axioms. A clustering function is rich if by modifying the distances any output can be obtained.

Definition 11 (Richness) A clustering function F satisfies richness if for any domain set X partitioned into that set, $X = X_1 \cup X_2, \dots, \cup X_n$, there exists a distance function d over X so that $F(X, d, n) = \{X_1, X_2, \dots, X_n\}$.

We propose an extension on richness. A clustering function satisfies extended richness if for every finite collection of disjoint domain sets (each with its own distance function), by setting the distances between the data sets, we can get F to output each of these data sets as a cluster. This corresponds to the intuition that if groups of points are moved sufficiently far apart, then they will be placed in separate clusters.

Definition 12 (Extended Richness) For every set of domains, $\{(X_1, d_1), \dots, (X_n, d_n)\}$, there exists a distance function \hat{d} over $\bigcup_{i=1}^n X_i$ that extends each of the d_i 's (for $i \leq n$), such that $F(\bigcup_{i=1}^n X_i, \hat{d}, n) = \{X_1, X_2, \dots, X_n\}$.

6 Main result

Our main result specifies properties of clustering functions that uniquely identify linkage-based clustering functions.

Theorem 13 A clustering function is linkage based if and only if it is hierarchical and it satisfies: Outer Consistency, Locality and Extended Richness.

We divide the proof into the following two sub-sections (one for each direction of the "if and only if").

6.1 The clustering function properties imply that the function is linkage-based

We show that if F satisfies the prescribed properties, then there exists a linkage function that, plugged into the procedure in the definition of a linkage-based function, will yield the same output as F (for every input (X, d) and k).

Lemma 14 If a clustering function F is hierarchical and it satisfies Outer Consistency, Locality and Extended Richness, then F is linkage-based.

The proof comprises the rest of this section.

Proof:

Since F is hierarchical, for every $1 \leq k < |X|$, $F(X, d, k)$ can be constructed from $F(X, d, k+1)$ by merging two clusters in $F(X, d, k+1)$. It remains to show that there exists a linkage function that determines which clusters to merge.

Due to the Isomorphism Invariance of F , one can assume, w.l.o.g., that the domain sets over which F is defined are (finite) subsets of the set of natural numbers, \mathcal{N} .

We define a relation over pairs of pairs of subsets $<_F$ and later prove that it is a (partial) ordering.

Definition 15 (The (pseudo-) partial ordering $<_F$) $<_F$ is a binary relation over equivalence classes, with respect to clustering-isomorphism. Namely $(A, B, d) \simeq (A', B', d')$ if the two pairs are isomorphic as clusters (see definition in the Preliminary section). We denote equivalence classes by square brackets. So, the domain of $<_F$ is

$$\{[A, B, d] : A \subseteq \mathcal{N}, B \subseteq \mathcal{N}, A \cap B = \emptyset \text{ and } d \text{ is a distance function over } A \cup B\}.$$

We define it by: $[(A, B, d)] <_F [(A', B', d')]$ if there exists a distance function d^* over $X = A \cup B \cup A' \cup B'$ that extends both d and d' (namely, $d \subseteq d^*$ and $d' \subseteq d^*$), and there exists $k \in \{2, 3\}$ such that

1. $A, B, A', B' \in F(X, d^*, k + 1)$
2. $A \cup B \in F(X, d^*, k)$
3. For all $D \in \{A, B, A', B'\}$, either $D \subseteq A \cup B$ or $D \in F(X, d^*, k)$.

Intuitively, $(A, B, d) <_F (A', B', d')$, if there is an input for which F creates the clusters A, B, A', B' as members of some clustering $F(X, d^*, k + 1)$, then $F(X, d^*, k)$ merges A with B (before it merges A' and B'). The relation is well defined thanks to the assumption that F is isomorphism invariant. For the sake of simplifying notation, we will omit the square brackets in the following discussion.

To show that for $<_F$ can be extended to a partial ordering, we first prove the following property.

Cycle freeness: Given a clustering function F that is outer-consistent, hierarchical, local and satisfies extended richness, there exists no finite sequence $(A_1, B_1, d_1) \dots (A_n, B_n, d_n)$, where $n > 2$, such that for all $1 \leq i < n$,

1. $A_i \cap B_i = \emptyset$,
2. d_i is a distance function over $A_i \cup B_i$ and
3. $(A_i, B_i, d_i) <_F (A_{i+1}, B_{i+1}, d_{i+1})$

and $(A_1, B_1, d_1) = (A_n, B_n, d_n)$.

This is shown in Lemma 17.

Next, we wish to show that for singleton sets $<_F$ respects the input distance function, d .

Lemma 16 For every x, y, x', y' , such that $x \neq y$ and $x' \neq y'$, every value $d_1(x, y)$ and $d_2(x', y')$, and every clustering function F ,

$$(\{x\}, \{y\}, d_1) <_F (\{x'\}, \{y'\}, d_2) \text{ if and only if } d_1(x, y) < d_2(x', y')$$

Proof:

Consider a data set on 4 points, $S = \{x, y, x', y'\}$. Let $a = d_1(x, y)$, $b = d_2(x', y')$. We construct a distance function d over S . Set $d(x, y) = b$ and $d(p, q) = a$ for all $\{p, q\} \neq \{x, y\}$.

Then either $(\{x\}, \{y\}, d_1) <_F (\{x'\}, \{y'\}, d_2)$ or $(\{x'\}, \{y'\}, d_2) <_F (\{x\}, \{y\}, d_1)$. Assume by way of contradiction that $d(x, y) < d'(x', y')$ but $(\{x'\}, \{y'\}, d_2) <_F (\{x\}, \{y\}, d_1)$. Then since $(\{x'\}, \{y'\}, d) <_F (\{x\}, \{y\}, d)$, $F(S, d, 3) = \{\{x, y\}, \{x'\}, \{y'\}\}$.

Set $c = b/a$. Note that $c > 1$. Let d' be such that $d'(x, y) = b$, $d'(x', y') = cb$, $d'(p, q) = a$ for all other pairs of elements in S . Then d' is an $(F(S, d, 3), d)$ -outer-consistent variant. Since F is outer-consistent, $F(S, d', 3) = F(S, d, 3)$. Next, consider the distance function d'' so that $d''(p, q) = (1/c) \cdot d'(p, q)$ for all $p, q \in S$. Since F is scale invariant, by condition 2 of Definition 2, $F(S, d'', 3) = F(S, d, 3)$. Finally, let d''' be such that $d'''(x', y') = b$, and $d'''(p, q) = a$ for all $\{p, q\} \neq \{x', y'\}$. Note that d''' is an $(F(S, d'', 3), d'')$ -outer-consistent variant. Therefore, $F(S, d''', 3) = F(S, d, 3) = \{\{x, y\}, \{x'\}, \{y'\}\}$. However, since $\ell(X_1, X_2, d_2) < \ell(X_1, X_2, d_1)$, $F(S, d''', 3) = \{\{x', y'\}, \{x\}, \{y\}\}$ - a contradiction. ■

Lemma 17 Given a clustering function F that is outer-consistent, hierarchical, local and satisfies extended richness, there exists no finite sequence $(A_1, B_1, d_1) \dots (A_n, B_n, d_n)$, where $n > 2$, such that for all $1 \leq i < n$,

1. $A_i \cap B_i = \emptyset$,
2. d_i is a distance function over $A_i \cup B_i$ and
3. $(A_i, B_i, d_i) <_F (A_{i+1}, B_{i+1}, d_{i+1})$

and $(A_1, B_1, d_1) = (A_n, B_n, d_n)$.

Proof:

Assume that such a sequence exists. Let $C_i = A_i \cup B_i$. and $X = \bigcup_{i=1}^n A_i \cup B_i$.

Using extended richness, we can construct \hat{d} from the given set of domains (C_i, d_i) , for all $1 \leq i \leq n$, that extends all of the distances, such that $F(X, \hat{d}, n) = \{C_1, C_2, \dots, C_n\}$.

Let us consider what happens for $F(X, \hat{d}, n+1)$. Since F is hierarchical, the $(n+1)$ -clustering must split one of the C_i 's. Given $1 \leq i < n$, we will show that you cannot split C_i without causing a contradiction.

Recall that $(A_i, B_i, d_i) <_F (A_{i+1}, B_{i+1}, d_{i+1})$, and thus there exists a distance function d' over $X' = A_i \cup B_i \cup A_{i+1} \cup B_{i+1}$, and $k \in \{2, 3\}$, such that $A_i, B_i, A_{i+1}, B_{i+1} \in F(X', d', k+1)$, $A_i \cup B_i \in F(X', d', k)$ and for all $D \in \{A_i, B_i, A_{i+1}, B_{i+1}\}$, either $D \subseteq A_i \cup B_i$ or $D \in F(X', d', k)$.

First, we will show that C_i must be split into A_i and B_i . Consider $F(C_i, d_i, 2)$. Since $(A_i, B_i, d_i) <_F (A_{i+1}, B_{i+1}, d_{i+1})$, we know that $F(C_i, d_i, 2) = \{A_i, B_i\}$, by locality.

Now we will show that splitting C_i into A_i and B_i violates $(A_i, B_i, d_i) <_F (A_{i+1}, B_{i+1}, d_{i+1})$. Using locality, we focus on the data points in $C_i \cup C_{i+1}$. By locality, for some $k \in \{2, 3\}$, $A_i, B_i \in F(C_i \cup C_{i+1}, \hat{d}/C_i \cup C_{i+1}, k)$. At this point, the distances defined by \hat{d} between C_i and C_{i+1} may be different from those defined in d' .

Using outer consistency, we define distance function \tilde{d} over X' that is both a $(F(C_i \cup C_{i+1}, \hat{d}/C_i \cup C_{i+1}, k), \hat{d}/C_i \cup C_{i+1})$ -outer consistent variant, and a $(F(C_i \cup C_{i+1}, d', k), d')$ -outer consistent variant.

First, let $m_1 = \max\{\hat{d}(x, y) \mid x, y \in C_i \cup C_{i+1}\}$ and let $m_2 = \max\{d'(x, y) \mid x, y \in C_i \cup C_{i+1}\}$. Finally, let $m^* = \max\{m_1, m_2\}$. Now, we defined \tilde{d} as follows:

$$\tilde{d}(x, y) = \begin{cases} \hat{d}(x, y) & \text{if } x, y \in C_i \text{ or } x, y \in C_{i+1} \\ m^* & \text{otherwise} \end{cases}$$

It is clear that \tilde{d} meets our requirements. By outer consistency, $F(C_i \cup C_{i+1}, \tilde{d}, k) = F(C_i \cup C_{i+1}, \hat{d}/C_i \cup C_{i+1}, k)$, in which we showed that A_i and B_i are separate clusters. Also by outer consistency, $F(C_i \cup C_{i+1}, \tilde{d}, k) = F(C_i \cup C_{i+1}, d', k)$, in which A_i and B_i are part of the same cluster by the ordering $<_F$. Thus, we have a contradiction because $C_i \neq C_{i+1}$. ■

Lemma 18 *If $R(,)$ is a binary relation over some domain D that satisfies antisymmetry and cycle-freeness then there exists a partial ordering $R^*(,)$ over D that extends R (i.e., for every $a, b \in D$, if $R(a, b)$ holds then so does $R^*(a, b)$). In fact, the transitive closure of R is such an extension.*

Let $<_F^*$ be the transitive closure of $<_F$. Applying the above lemma it is a partial ordering. The next step is to use the partial ordering $<_F^*$ to define a linkage function that demonstrates that F is a linkage-based clustering.

We shall apply the following basic universality result for partial orderings:

Lemma 19 *Let \prec be a partial ordering over some finite or countable set D , and let h be an order preserving mapping of some $D' \subseteq D$ into the positive reals³, then there exist an extension of h , $\hat{h} : D \rightarrow \mathcal{R}^+$ that is order preserving.*

Finally, we define the embedding

$$\ell_F : \{(A, B, d) \mid A \subseteq \mathcal{N}, B \subseteq \mathcal{N}, A \cap B = \emptyset \text{ and } d \text{ is a distance function over } A \cup B\} \rightarrow \mathcal{R}^+$$

by applying Lemma 19 to $<_F^*$.

There is one potential inconsistency between ℓ_F and property 2 in Definition 3. We now show how we can modify ℓ_F so that property 2 is satisfied. Say that there exists some d_1 over $X_1 \cup X_2$ where d_2 is an $(\{X_1, X_2\}, d_1)$ -outer-consistent variant so that $\ell_F(X_1, X_2, d_2) < \ell_F(X_1, X_2, d_1)$. That is, if we move X_1 and X_2 further from each other then the ℓ_F value decreases. By Lemma 20, there does not exist (Y_1, Y_2, d_3) that is assigned an ℓ_F value that is strictly between $\ell_F(X_1, X_2, d_1)$ and $\ell_F(X_1, X_2, d_2)$. In addition, $<_F$ does not specify a relationship between (X_1, X_2, d_2) and (X_1, X_2, d_1) by definition of $<_F$, since a function cannot extend both d_1 and d_2 simultaneously unless $d_1 = d_2$. Therefore, we can modify ℓ_F so that $\ell(X_1, X_2, d_1) \leq \ell(X_1, X_2, d_2)$ and so property 2 of Definition 3 holds.

Lemma 20 *Let X_1 and X_2 be some domains, distance d_1 over $X_1 \cup X_2$, and d_2 an $(\{X_1, X_2\}, d_1)$ -outer-consistent variant so that $\ell_F(X_1, X_2, d_2) < \ell_F(X_1, X_2, d_1)$. Then there do not exist Y_1, Y_2 , and d_3 so that $\ell_F(X_1, X_2, d_2) < \ell_F(Y_1, Y_2, d_3) < \ell_F(X_1, X_2, d_1)$.*

³any dense linear ordering with no first element and no last element has the same universality property

Proof: Assume by way of contradiction that such Y_1 , Y_2 , and d_3 exist. Then there exists a distance function d^* over $X^* = X_1 \cup X_2 \cup Y_1 \cup Y_2$, $d_1, d_3 \subseteq d^*$, so that F on (X^*, d^*) merges Y_1 and Y_2 before X_1, X_2 . If we move X_1 and X_2 away from each other, so that d_2 would describe the distances in $X_1 \cup X_2$, then X_1 and X_2 would be merged before Y_1 and Y_2 . This contradicts outer consistency. Therefore, there are no values of ℓ_F between $\ell_F(X_1, X_2, d_2)$ and $\ell_F(X_1, X_2, d_1)$. ■

Lemma 21 *The function ℓ_F is a linkage function for any hierarchical function F that satisfies locality, outer-consistency, and extended richness.*

Proof: ℓ_F satisfies condition 1 of Definition 3 since it is defined on equivalence classes of isomorphic sets. The function ℓ_F satisfies condition 2 of Definition 3 by construction (by the modification explained above). By Lemma 22 ℓ_F satisfied condition 3 in Definition 3. ■

Claim 22 *The function ℓ_F , for any hierarchical function F that satisfies locality, outer-consistency, and extended richness, satisfies condition 3 of Definition 3.*

Proof: Let r be in the range of ℓ_F . Then there exist data sets (X_3, d_3) and (X_4, d_4) , $X_3 \cap X_4 = \emptyset$, and distance d' over $X_3 \cup X_4$, such that $\ell_F(X_3, X_4, d') \geq r$. Let $(X_1, d_1), (X_2, d_2)$ be a pair of data sets as defined above. If $\{X_1, X_2\} = \{X_3, X_4\}$ then we are done, so assume that $\{X_1, X_2\} \neq \{X_3, X_4\}$.

By extended richness, there exists a distance function \hat{d} over $X = \bigcup X_i$ that extends d_1, d_2, d_3, d_4 such that $F(X, \hat{d}, 4) = \{X_1, X_2, X_3, X_4\}$. We define \tilde{d} to be an $(F(X, \hat{d}, 4), \hat{d})$ -outer consistent variant defined as follows:

$\tilde{d}(x, y) = \max\{\hat{d}(x, y), d'(x, y)\}$ when $x \in X_3, y \in X_4$ or $x \in X_4, y \in X_3$ and $\tilde{d}(x, y) = \hat{d}(x, y)$ otherwise.

Notice that $\tilde{d}/X_3 \cup X_4$ is an $(F(X_3 \cup X_4, d', 2), d')$ -outer consistent variant. Thus, $\ell_F(X_3, X_4, \tilde{d}/X_3 \cup X_4) \geq r$.

Also by extended richness, there exists a distance function \hat{d}' over X that extends $d_1, d_2, \tilde{d}/X_3 \cup X_4$ such that $F(X, \hat{d}', 3) = \{X_1, X_2, X_3 \cup X_4\}$. Using outer consistency, we can find \tilde{d}' that is an $(F(X, \hat{d}, 4), \hat{d})$ -outer consistent variant and an $F(X, \hat{d}', 3), \hat{d}'$ -outer consistent variant by just increasing distances between X_i and X_j , where $i \neq j$ and $\{i, j\} \neq \{3, 4\}$. Thus, $F(X, \tilde{d}', 4) = \{X_1, X_2, X_3, X_4\}$ and $F(X, \tilde{d}', 3) = \{X_1, X_2, X_3 \cup X_4\}$. Therefore, $\ell_F(X_1, X_2, \tilde{d}') > \ell_F(X_3, X_4, \tilde{d}) \geq r$. ■

Claim 23 *For every clustering function F , the linkage-based clustering that ℓ_F defines agrees with F on any input data set.*

Proof: For every (X, d) , the linkage based clustering that ℓ_F defines starts with the clusters consisting of all singletons, and at each step merges two clusters. Thus, for all $2 \leq k \leq |X|$, we have a k -clustering C and the $k-1$ clustering merges some $C_1, C_2 \in C$, where $C_1 \cup C_2 = C$ or $\ell_F(C_1, C_2) < \ell_F(C_3, C_4)$, for all $C_3, C_4 \in C$, $\{C_3, C_4\} \neq \{C_1, C_2\}$. Therefore, for all $2 \leq k \leq |X|$, $(C_1, C_2, d/C_1 \cup C_2) <_F (C_3, C_4, d/C_3 \cup C_4)$, for all C_3, C_4 as described, by our construction of ℓ_F . Therefore, F would merge the same clusters to obtain the $k-1$ clustering, and so ℓ_F agrees with F for any input (X, d) on all k -clusterings, $2 \leq k \leq |X|$. Clearly they also agree when $k = 1$. ■

This concludes the proof of Lemma 14.

6.2 Every linkage-based clustering function is hierarchical, local, and outer-consistent

If a clustering function is linkage-based, then by construction it is hierarchical.

Lemma 24 *Every linkage-based clustering function is hierarchical.*

Proof: For every $1 \leq k' \leq k \leq |X|$, by definition of linkage based, $F(X, d, k)$ can be constructed from $F(X, d, k')$ by continually merging clusters until k' clusters remain. ■

Lemma 25 *Every linkage-based clustering function F is local.*

Proof: Let k' -clustering C be a subset of $F(X, d, k)$. Let $X' = \bigcup_{c \in C} c$.

We will show that for all $k' \leq i \leq |X'|$, $F(X', d/X', i)$ is a subset of $F(X, d, j)$ for some j . After, we conclude our proof using the following argument: $F(X', d/X', k')$ has k' clusters, $F(X', d/X', k')$ is a subset of $F(X, d, j)$ for some j , and since between $F(X, d, j)$ and $F(X, d, k)$ in the algorithm we cannot merge clusters in C (as C would no longer be a subset of $F(X, d, k)$), this gives us that $F(X', d/X', k')$ is a subset of $F(X, d, k)$ and it is equal to C .

The base case follows from the observation that $F(X', d/X', |X'|)$ and $F(X, d, |X|)$ both consist of singleton clusters.

For some $i > k'$, assume that there exists a j such that $F(X', d/X', i)$ is a subset of $F(X, d, j)$. We need to show that there exists a j' such that $F(X', d/X', i - 1)$ is a subset of $F(X, d, j')$.

Since F is linkage based, there exists a linkage function ℓ so that when ℓ is used in the algorithm in Definition 4, the algorithm yields the same output as F .

Since $F(X', d/X', i) \subseteq F(X, d, j)$, and $C \subseteq F(X, d, k)$, there exists a j^* so that $F(X, d, j^*)$ can be obtained from $F(X, d, j^* + 1)$ by merging two clusters in $F(X, d, j) \cap C$. The pair of clusters with minimal ℓ value in $F(X, d, j) \cap C$ is the same as the pair of clusters with minimal ℓ value in $F(X', d/X', i)$. Therefore, $j' = j^*$. ■

Lemma 26 *Every linkage-based clustering function F is outer-consistent.*

Proof: By the monotonicity condition in Definition 3, whenever two clusters are pulled further apart from each other, the corresponding ℓ value does not decrease. Consider some data set (X, d) and d' an $(F(X, d, k), d)$ -outer-consistent variant. We will show that $F(X, d, k) = F(X, d', k)$ by induction on k . Clearly, $F(X, d, |X|) = F(X, d', |X|)$. Assume that $F(X, d, j) = F(X, d', j)$ for some $j > k$. In order to obtain $F(X, d', j - 1)$, F merges the pair of clusters $c'_1, c'_2 \in F(X, d', j)$ with minimal ℓ value. Similarly, to obtain $F(X, d, j - 1)$, F merges the pair $c_1, c_2 \in F(X, d, j)$.

Suppose that $\{c_1, c_2\} \neq \{c'_1, c'_2\}$. Then $\ell(c'_1, c'_2, d) \leq \ell(c'_1, c'_2, d') < \ell(c_1, c_2, d') = \ell(c_1, c_2, d)$, where the first equality follows by monotonicity and the second inequality follows by the minimality of $\ell(c'_1, c'_2, d')$. Note that $c_1, c_2 \subseteq C_k$, where $C_k \in F(X, d, k)$. That is, c_1 and c_2 are part of the same cluster in $F(X, d, k)$, and since d' is an $(F(X, d, k), d)$ -outer-consistent variant, the equality follows by representation-independence. But $\ell(c'_1, c'_2, d) < \ell(c_1, c_2, d)$ contradicts the minimality of $\ell(c_1, c_2, d)$, so $\{c_1, c_2\} = \{c'_1, c'_2\}$. ■

Lemma 27 *Every linkage-based function satisfies extended richness.*

Proof: Let $(X_1, d_1), (X_2, d_2), \dots, (X_n, d_n)$ be some data sets. We will show that there exists an extension d of d_1, d_2, \dots, d_n so that $F(\bigcup_{i=1}^n X_i, d, n) = \{X_1, X_2, \dots, X_n\}$.

To make F give this output, we design d in such a way that for any i , and $A, B \subseteq X_i$, and any $C \subseteq X_i$, and $D \subseteq X_j$ where $i \neq j$, $\ell(A, B, d) < \ell(C, D, d)$.

Let $r = \max_{i, A, B \subseteq X_i, i \in \{1, 2\}} \ell(A, B)$. Since ℓ satisfies property 4 of Definition 3, for any $C \subseteq X_i$, $D \subseteq X_j$, for $i \neq j$, there exists a distance function d_{CD} that extends d_i/C and d_j/D so that $\ell(C, D) > r$. Consider constructing such distance function d_{CD} for every pair $C \subseteq X_i$ and $D \subseteq X_j$, where $i \neq j$. Then, let $m = \max_{i \neq j, C \subseteq X_i, D \subseteq X_j} \max_{x \in C, y \in D} d_{CD}(x, y)$.

We define d as follows: $d(x, y) = d_i(x, y)$ if $x, y \in X_i$ for some i and $d(x, y) = m$ otherwise. Since ℓ satisfies property 2 of Definition 3, $\ell(C, D) > r$ for all $C \in X_i, D \in X_j$ where $i \neq j$. On the other hand, $\ell(A, B) \leq r$ for any $A, B \subseteq X_i$ for some i . Therefore, the algorithm will not merge any $C \subseteq X_i$ with $D \subseteq X_j$ where $i \neq j$, while there is any clusters $A, B \subseteq X_i$ for some i remaining. This gives that $F(\bigcup_{i=1}^n X_i, d, n) = \{X_1, X_2, \dots, X_n\}$. ■

Finally, we put our results together to conclude the main theorem.

Theorem 13 restated *A clustering function is linkage based if and only if it is hierarchical and it satisfies: Outer Consistency, Locality and Extended Richness.*

Proof: By Theorem 4, if a clustering function is outer-consistent, hierarchical, and local, then it is linkage-based. By Lemma 24, every linkage-based clustering function is hierarchical. By and Lemma 25 every linkage-based clustering function is local. By Lemma 26, every linkage-based clustering function is outer-consistent. Finally, by Lemma 27, every linkage based function satisfies extended richness. ■

7 Relaxations of a Linkage Functions and Corresponding Characterizations

7.1 Simplified linkage function

Our proof also yields some insights about clustering that are defined by looser notions of linkage functions. We describe the characterization of the class of clustering functions that are based of linkage functions that are not required to obey the conditions of Definition 3.

Definition 28 (Simplified linkage function) A linkage function ℓ takes a data set (X, d) and a partition (X_1, X_2) of the domain X .

We then define a *simplified linkage-based function* as in Definition 4, but with a simplified linkage function instead of the linkage function in Definition 3.

We obtain an interesting characterization of simplified linkage-based function that satisfy outer-consistency and extended richness.

Theorem 29 A clustering function that satisfies outer-consistency and extended richness is simplified linkage-based if and only if it hierarchical and local.

Proof: Since a linkage function is a simplified linkage function with additional constraints, by Theorem 14 we get that an outer-consistent, hierarchical and local clustering function is simplified linkage-based. The results and proofs of Theorem 24 and Theorem 25 also apply for simplified linkage functions, thus showing that simplified linkage-based functions are hierarchical and local. ■

7.2 General linkage function

Unlike linkage-based clustering functions defined in Definition 4 or simplified linkage-based functions, a *general linkage-based clustering function* might use a different linkage procedure on every data set.

This results from a modification on the definition of a linkage function, allowing the function to have access to the entire data set, outside the two clusters under comparison.

Definition 30 (General linkage function) A general linkage function is given a data set (X, d) and $A, B \subseteq X$, and outputs a real number.

Note that in the above definition, A and B need not partition X . As such, the function may use information outside of both A and B to determine what value to assign to this pair of clusters. We define a *general linkage-based clustering function* as in Definition 4, except using a general linkage function instead of the linkage function in definition 3.

Definition 31 (general linkage-based clustering function) A clustering function F is linkage-based if there exists a general linkage function ℓ so that

- $F(X, d, |X|) = \{\{x\} \mid x \in X\}$
- For $1 \leq k < |X|$, $F(X, d, k)$ is constructed by merging the two clusters in $F(X, d, k+1)$ that minimize the value of \bar{d} . Formally,

$$F(X, d, k) = \{c \mid c \in F(X, d, k+1), c \neq c_i, c \neq c_j\} \cup \{c_i \cup c_j\},$$

such that $\{c_i, c_j\} = \operatorname{argmin}_{\{c_i, c_j\} \subseteq F(X, d, (k+1))} \ell((X, d), c_i, c_j)$.

For example, a clustering function that uses single-linkage on data sets with an even number of points, and maximal linkage on data sets with an odd number of points, is not linkage-based, but it is a general linkage-based clustering function. Many other examples of general linkage-based functions are artificial, and do not correspond to what is commonly thought of as linkage-based clustering. Yet general linkage-based functions include linkage-based functions, and are actually easier to characterize.

Theorem 32 A clustering function is hierarchical if and only if it is a general linkage-based clustering function.

Proof: For every $1 \leq k \leq k' \leq |X|$, by definition of a general linkage-based clustering function, $F(X, d_X, k)$ can be constructed from $F(X, d_X, k')$ by continually merging clusters until k clusters remain. Therefore, general linkage-based functions are hierarchical.

Assume that F hierarchical. Then whenever $k' > k$, $F(X, d, k)$ can be obtained from $F(X, d, k')$ by merging clusters in $F(X, d_X, k')$. In particular, $F(X, d, k)$ can be obtained from $F(X, d, k+1)$ by merging

a pair of clusters in $F(X, d, k + 1)$. It remains to show that there exists a general linkage function ℓ that defines which clusters are merged.

We now show how to construct the general linkage function. For every (X, d) , and for every k , if $F(X, d, k)$ can be obtained from $F(X, d, k + 1)$ by merging clusters a and b , then set $\ell((X, d), (A, B)) = |X| - k$. For the remaining $a, b \subseteq X$, set $\ell((X, d)(a, b)) = |X|$.

Consider the the function resulting from using the general linkage function ℓ to determine which pair of clusters to merge, until k clusters remain. Clearly, $F'(X, d, |X|) = F(X, d, |X|)$. Assume that $F'(X, d, k + 1) = F(X, d, k + 1)$. We show that $F'(X, d, k) = F(X, d, k)$. Since F' is a general linkage based clustering function, it merges some clusters $c_1, c_2 \in F'(X, d, k + 1)$ to obtain $F'(X, d, k)$. Since F is hierarchical, it merges some clusters $c_3, c_4 \in F(X, d, k + 1)$ to obtain $F(X, d, k)$, therefore $\ell((X, d)(c_3, c_4)) = |X| - k$. For any $\{c_5, c_6\} \in F(X, d, k)$ so that $\{c_5, c_6\} \neq \{c_3, c_4\}$, either c_5 and c_6 are merged to obtain $F(X, d, k')$ for some $k' < k$ and so $\ell((X, d)(c_5, c_6)) = |X| - k'$, or c_5 and c_6 are never merged directly (they are first merged with other clusters), and so $\ell((X, d)(c_5, c_6)) = |X|$. In either case, $\ell((X, d)(c_3, c_4)) < \ell((X, d)(c_5, c_6))$. Since ℓ defines F' , F' merges $c_1, c_2 \in F'(X, d, k + 1) = F(X, d, k + 1)$ to obtain $F'(X, d, k)$. Therefore, $\{c_1, c_2\} = \{c_3, c_4\}$ and so $F'(X, d, k) = F(X, d, k)$. ■

8 Conclusions

We address the task of understanding the consequences of choosing one clustering algorithm over another. As we have argued in the introduction, we view it as a very important task, if only to help users make better educated choices about the clustering algorithms that they apply to their data. In spite of its importance, very little has been previously done along these lines.

In any attempt to taxonomize clusterings, a natural family of clustering that one needs to distinguish is the class of linkage based clustering. In this work we succeeded in characterizing this rich and popular class by a relatively small set of intuitive abstract properties. We sincerely believe that these results will encourage other researchers to follow up this ambitious task of providing guidelines in the zoo of clustering algorithms.

9 Acknowledgements

We wish to thank Reza Bosagh Zadeh for stimulating discussions of the axiomatic approach to clustering and for proposing the locality property in that context.

References

- [1] M. Ackerman and S. Ben-David. Measures of Clustering Quality: A Working Set of Axioms for Clustering. NIPS, 2008.
- [2] Reza Bosagh Zadeh and Shai Ben-David. "A Uniqueness Theorem for Clustering." The 25th Annual Conference on Uncertainty in Artificial Intelligence (UAI 09), 2009.
- [3] Chris Ding and Xiaofeng He. "Cluster Aggregate Inequality and Multi-level Hierarchical Clustering." Knowledge Discovery in Databases (PKDD), 2005. LNCS, Springer Berlin / Heidelberg, V. 3721, pp. 71-83
- [4] Brian Everitt, Sabine Landau, and Morven Leese. Cluster Analysis, 4th edn. Arnold, London. 2001.
- [5] Derek Greene, Gerard Cagney, Nevan Krogan, and Pdraig Cunningham. "Ensemble non-negative matrix factorization methods for clustering proteinprotein interactions." Bioinformatics Vol. 24 no. 15 2008, pages 1722-1728
- [6] N. Jardine and R. Sibson. The construction of hierarchic and non-hierarchic classifications. Multivariate Statistical Methods, Among-groups Covariation, 1975.
- [7] Jon Kleinberg. "An Impossibility Theorem for Clustering." Advances in Neural Information Processing Systems (NIPS) 15, 2002.
- [8] U. von Luxburg. A Tutorial on Spectral Clustering. Statistics and Computing 17(4): 395-416, 2007

10 Appendix: spectral clustering does not satisfy locality

In this appendix we show that spectral clustering does not satisfy locality. This illustrates that locality is a property of clustering functions (one that is satisfied by some, but not all, reasonable clustering functions), and not an axiom (which is satisfied by all reasonable clustering functions).

We discuss two clustering functions from spectral clustering: ratio-cut and normalized-cut. For more on spectral clustering, see a tutorial by Luxburg [8].

In spectral clustering we assume that there is an underlying similarity function s , instead of a distance function. The only difference between a distance functions and a similarity functions is that higher values of represent greater similarity when using similarity functions, while the opposite holds for distance functions.

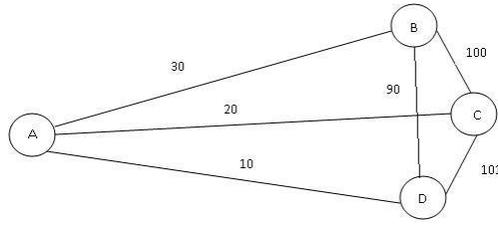


Figure 1: A data set used to illustrate that Ratio-Cut does not satisfy locality.

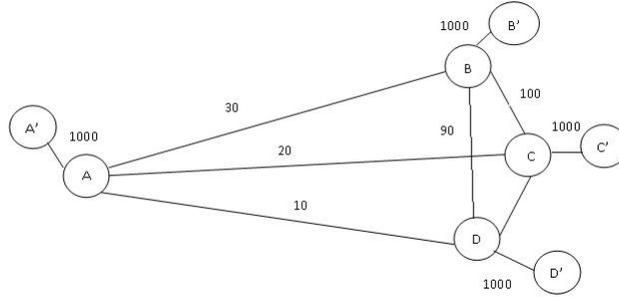


Figure 2: A data set used to illustrate that Ratio-Cut does not satisfy locality.

Let $|c|$ denote the number of elements in cluster c . \bar{c} denotes the data set X without the points in c . For $c_1, c_2 \subseteq X$, let $cut(c_1, c_2) = \sum_{a \in c_1, b \in c_2} s(a, b)$.

Definition 33 (ratio-cut clustering function) Given a data set (X, d) and an integer $1 \leq k \leq |X|$, the ratio-cut clustering function finds a k -clustering $\{c_1, c_2, \dots, c_k\}$ that minimizes

$$\sum_1^k \frac{cut(c_i, \bar{c}_i)}{|c_i|}.$$

The normalized-cut clustering function takes into account within-cluster similarity. Let $vol(c_i) = \sum_{a, b \in c_i} s(a, b)$.

Definition 34 (normalized-cut clustering function) Given a data set (X, d) and an integer $1 \leq k \leq |X|$, the normalized-cut clustering function finds a k -clustering $\{c_1, c_2, \dots, c_k\}$ that minimizes

$$\sum_1^k \frac{cut(c_i, \bar{c}_i)}{vol(c_i)}.$$

Theorem 35 *Ratio-Cut is not local.*

Proof:

Figure 1 illustrates a data set (with the similarity indicated on the arrows) where the optimal ratio-cut 3-clustering is $\{\{A\}, \{B, C\}, \{D\}\}$. However, on data set $\{B, C, D\}$ (with the same pairwise similarities as in Figure 1), the clustering $\{\{B\}, \{C, D\}\}$ has lower ratio-cut than $\{\{B, C\}, \{D\}\}$. ■

We now show that normalized-cut is not local.

Theorem 36 *Normalized-Cut is not local.*

Proof: Figure 2 illustrates a data set with the similarities indicated on the arrows - a missing arrow indicates a similarity of 0. The optimal normalized-cut 3-clustering is $\{\{A, A'\}, \{B, B', C, C'\}, \{D, D'\}\}$. However, on data set $\{B, B', C, C', D, D'\}$ (with the same pairwise similarity as in Figure 2), the clustering $\{\{B, B'\}, \{C, C', D, D'\}\}$ has lower normalized-cut than $\{\{B, B', C, C'\}, \{D, D'\}\}$. ■

Robust Hierarchical Clustering

Maria Florina Balcan
Georgia Institute of Technology
School of Computer Science
ninamf@cc.gatech.edu

Pramod Gupta
Georgia Institute of Technology
School of Computer Science
pramodgupta@gatech.edu

Abstract

One of the most widely used techniques for data clustering is agglomerative clustering. Such algorithms have been long used across many different fields ranging from computational biology to social sciences to computer vision in part because their output is easy to interpret. Unfortunately, it is well known, however, that many of the classic agglomerative clustering algorithms are *not* robust to noise [14]. In this paper we propose and analyze a new robust algorithm for bottom-up agglomerative clustering. We show that our algorithm can be used to cluster accurately in cases where the data satisfies a number of natural properties and where the traditional agglomerative algorithms fail. We also show how to adapt our algorithm to the inductive setting where our given data is only a small random sample of the entire data set.

1 Introduction

Many data mining and machine learning applications ranging from computer vision to biology problems have recently faced an explosion of data. As a consequence it has become increasingly important to develop effective, accurate, robust to noise, fast, and general clustering algorithms, accessible to developers and researchers in a diverse range of areas.

One of the oldest and most commonly used methods for clustering data, widely used in many scientific applications, is hierarchical clustering [5, 6, 8, 10, 7, 11, 12, 14, 9, 13, 15]. In hierarchical clustering the goal is not to find a single partitioning of the data, but a hierarchy (generally represented by a tree) of partitions which may reveal interesting structure in the data at multiple levels of granularity. The most widely used hierarchical methods are the agglomerative clustering techniques; most of these techniques start with a separate cluster for each point and then progressively merge the two closest clusters until only a single cluster remains. In all cases, we assume that we have a measure of similarity between pairs of objects, but the different schemes are distinguished by how they convert this into a measure of similarity between two clusters. For example, in single linkage the similarity between two clusters is the maximum similarity between points in these two different clusters. In complete linkage, the similarity between two clusters is the minimum similarity between points in these two different clusters. Average linkage has various variants, for example, a common one defines the similarity between two clusters as the average similarity between points in these two different clusters [7, 12].

Such algorithms have been used in a wide range of application domains ranging from biology applications to social sciences to computer vision applications mainly because they are quite fast and the output is quite easy to interpret. It is well known, however, that one of the main limitations of the agglomerative clustering algorithms is that they are *not* robust to noise [14]. In this paper we propose and analyze a robust algorithm for bottom-up agglomerative clustering. We show that our algorithm satisfies formal robustness guarantees and it will be successful in cases where the traditional agglomerative algorithms fail.

In order to formally analyze correctness of our algorithm we use the framework introduced by Balcan et. al [2]. In this framework, we assume there is some target clustering (much like a k -class target function in the multi-class learning setting) and we say that an algorithm correctly clusters data satisfying property P if on any data set having property P , the algorithm produces a tree such that the target is some pruning of the tree. For example if all points are more similar to points

in their own target cluster than to points in any other cluster (this is called the strict separation property), then any of the standard agglomerative algorithms will succeed. See Figure 1. However, with just tiny bit of noise, for example if each point has even just one point from a different cluster that it is similar too, then the standard algorithms will all fail (we elaborate on this in Section 2.2). See Figure 2. This brings up the question: is it possible to design an agglomerative algorithm that is robust to these types of situations and can tolerate a substantial degree of noise? The contribution of our paper is to provide a positive answer to this question; we develop a robust, linkage based algorithm that will succeed in interesting cases where standard agglomerative algorithms will fail.

At a high level, our new algorithm is robust to noise in two different and important ways. First, it uses more global information for creating an interesting starting point for a linkage procedure, a set of not too small, but also not too large blobs that are mostly “pure” (these blobs are created by grouping together vertices with a lot of neighbors in common); second, it uses a robust linkage procedure (which is based on a score involving the *median*) for merging large enough blobs. Using blobs and the median lend the robustness, since, roughly speaking, noisy similarities are outvoted.

1.1 Our Results

We present a new and robust algorithm for agglomerative clustering and we show that our algorithm will be successful in many cases where standard agglomerative algorithms will fail.

In particular, we show that if the data satisfies a natural good neighborhood property, then our algorithm can be used to cluster well in the tree model (i.e., to output a hierarchy such that the target clustering is a pruning of that hierarchy). The good neighborhood property roughly says after a small number of malicious points have been removed, for the remaining points, most of their nearest neighbors are from their target cluster. We also show how to adapt our algorithm to the inductive setting, where our given data is only a small random sample of the entire data set. Based on such a sample, our algorithm outputs an implicit hierarchy of clusterings of the full domain, that is evaluated with respect to the underlying distribution. A nice property of the condition and of the algorithm we analyze is that they are insensitive to any monotone transformation of the similarities.

It is worth noting that the good neighborhood property is much broader than the ν -strict separation property, a generalization of the simple strict separation property discussed above, requiring that after a small number of outliers have been removed all points are strictly more similar to points in their own cluster than to points in other clusters. Balcan et. al [2] also analyzed the ν -strict separation condition and provided an algorithm for producing a hierarchy with the desired property, but via a much more computationally expensive (non-agglomerative) algorithm. Our algorithm is simpler, faster, and much more generally applicable compared to the algorithm in [2] specifically designed for ν -strict separation.

1.2 Related Work

In agglomerative hierarchical clustering [8, 7, 11, 12] the goal is not to find a single partitioning of the data, but a hierarchy (generally represented by a tree) of partitionings which may reveal interesting structure in the data at multiple levels of granularity. Traditionally, only clusterings at a certain level are considered, but as we argue in Section 2 it is more desirable to consider all the prunings of the tree, since this way we can then handle much more general situations. As mentioned above, it is well known that standard agglomerative hierarchical clustering techniques are not tolerant to noise.

2 Definitions. A Formal Setup

We consider a clustering problem (S, ℓ) specified as follows. Assume we have a data set S of n objects. Each $x \in S$ has some (unknown) “ground-truth” label $\ell(x)$ in $Y = \{1, \dots, k\}$, where we will think of k as much smaller than n . We let $C_i = \{x \in S : \ell(x) = i\}$ denote the set of points of label i (which could be empty), and denote the target clustering as $\mathcal{C} = \{C_1, \dots, C_k\}$. Given another proposed clustering $h, h : S \rightarrow Y$, we define the error of h with respect to the target clustering to be the fraction of points on which h and \mathcal{C} disagree under the optimal matching of clusters in h to clusters in \mathcal{C} ; i.e.,

$$err(h) = \min_{\sigma \in \mathcal{S}_k} \left[\Pr_{x \in S} [\sigma(h(x)) \neq \ell(x)] \right],$$

where \mathcal{S}_k is the set of all permutations on $\{1, \dots, k\}$. Equivalently, the error of a clustering $\mathcal{C}' = \{C'_1, \dots, C'_k\}$ can be expressed as

$$\min_{\sigma \in \mathcal{S}_k} \frac{1}{n} \sum_i |C_i - C'_{\sigma(i)}|.$$

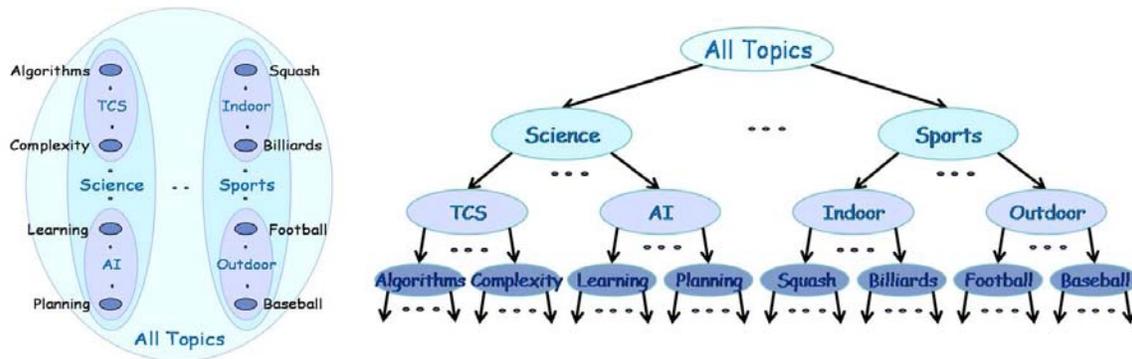


Figure 1: Consider a document clustering problem. Assume that data lies in multiple regions Algorithms, Complexity, Learning, Planning, Squash, Billiards, Football, Baseball. Suppose that $\mathcal{K}(x, y) = 0.999$ if x and y belong to the same inner region; $\mathcal{K}(x, y) = 3/4$ if $x \in$ Algorithms and $y \in$ Complexity, or if $x \in$ Learning and $y \in$ Planning, or if $x \in$ Squash and $y \in$ Billiards, or if $x \in$ Football and $y \in$ Baseball; $\mathcal{K}(x, y) = 1/2$ if x is in (Algorithms or Complexity) and y is in (Learning or Planning), or if x is in (Squash or Billiards) and y is in (Football or Baseball); define $\mathcal{K}(x, y) = 0$ otherwise. Both clusterings $\{\text{Algorithms} \cup \text{Complexity} \cup \text{Learning} \cup \text{Planning}, \text{Squash} \cup \text{Billiards}, \text{Football} \cup \text{Baseball}\}$ and $\{\text{Algorithms} \cup \text{Complexity}, \text{Learning} \cup \text{Planning}, \text{Squash} \cup \text{Billiards} \cup \text{Football} \cup \text{Baseball}\}$ satisfy the strict separation property.

We will be considering clustering algorithms whose only access to their data is via a pairwise similarity function $\mathcal{K}(x, x')$ that given two examples outputs a number in the range $[-1, 1]$. We will say that \mathcal{K} is a symmetric similarity function if $\mathcal{K}(x, x') = \mathcal{K}(x', x)$ for all x, x' . In this paper we assume that the similarity function \mathcal{K} is symmetric. For $A \subseteq S$, we denote by n_A the number of points in A .

Our goal is to produce a hierarchical clustering that contains a pruning that is close to the target clustering. Formally, the goal of the algorithm is to produce a hierarchical clustering: that is, a tree on subsets such that the root is the set S , and the children of any node S' in the tree form a partition of S' . The requirement is that there must exist a *pruning* h of the tree (not necessarily using nodes all at the same level) that has error at most ϵ . Balcan et. al [2] have shown that this type of output is necessary in order to be able to analyze non-trivial properties of the similarity function. For example, even if the similarity function satisfies the requirement that all points are more similar to all points in their own cluster than to any point in any other cluster (this is called the strict separation property) and even if we are told the number of clusters, there can still be multiple different clusterings that satisfy the property. In particular, one can show examples of similarity functions and two significantly different clusterings of the data satisfying the strict separation property. See Figure 1 for an example. However, under the strict separation property, there is a single hierarchical decomposition such that any consistent clustering is a pruning of this tree. This motivates clustering in the tree model and this is the model we consider in this work as well.

Given a similarity function satisfying the strict separation property (see Figure 1 for an example), we can efficiently construct a tree such that the ground-truth clustering is a pruning of this tree [2]. Moreover, almost any of the standard linkage based algorithms (*e.g.*, single linkage, average linkage, or complete linkage) would work well under this property. However, one can show that if the similarity function slightly deviates from the strict separation condition, then all the standard agglomerative algorithms will fail (we elaborate on this in section 2.2). In this context, the main question we address in this work is: Can we develop other more robust, linkage based algorithms that will succeed under more realistic and yet natural conditions on the similarity function?

Note: Note that strict separation does not guarantee that all the cutoffs for different points x are the same, so single linkage would not necessarily have the right clustering if just stopped once it has k clusters; however the target clustering will probably be a pruning of the final single linkage tree; this is why we define success based on prunings.

2.1 Properties of the similarity function

We describe here some natural properties of the similarity functions that we analyze in this paper. We start with a noisy version of the simple strict separation property (mentioned above) which was introduced in [2] and we then define an interesting and natural generalization of it.

Property 1 *The similarity function \mathcal{K} satisfies ν -strict separation for the clustering problem (S, ℓ) if for some $S' \subseteq S$ of size $(1 - \nu)n$, \mathcal{K} satisfies strict separation for (S', ℓ) . That is, for all $x, x', x'' \in S'$ with $x' \in C(x)$ and $x'' \notin C(x)$ we have $\mathcal{K}(x, x') > \mathcal{K}(x, x'')$.*

So, in other words we require that the strict separation is satisfied after a number of bad points have been removed. A somewhat different condition is to allow each point to have some bad immediate neighbors as long as most of its immediate neighbors are good. Formally:

Property 2 *The similarity function \mathcal{K} satisfies α -good neighborhood property for the clustering problem (S, ℓ) if for all points x we have that all but αn out of their $n_{C(x)}$ nearest neighbors belong to the cluster $C(x)$.¹*

Note that α -good neighborhood property is different from the ν -strict separation property. For the ν -strict separation property we can have up to νn bad points that can misbehave; in particular, these νn bad points can have similarity 1 to *all* the points in S ; however, once we remove these points the remaining points are more similar to points in their own cluster than to points in other cluster. On the other hand, for the α -good neighborhood property we require that for all points x all but αn out of their $n_{C(x)}$ nearest neighbors belong to the cluster $C(x)$. (So we cannot have a point that has similarity 1 to all the points in S .) Note however that different points might misbehave on different αn neighbors. We can also consider a property that generalizes both the ν -strict separation property and the α -good neighborhood. Specifically:

Property 3 *The similarity function \mathcal{K} satisfies (α, ν) -good neighborhood property for the clustering problem (S, ℓ) if for some $S' \subseteq S$ of size $(1 - \nu)n$, \mathcal{K} satisfies α -good neighborhood property for (S', ℓ) . That is, for all for all points $x \in S'$ we have that all but αn out of their $n_{C(x) \cap S'}$ nearest neighbors in S' belong to the cluster $C(x)$.*

It is easy to see that:

Fact 1 *If the similarity function \mathcal{K} satisfies the α -good neighborhood property for the clustering problem (S, ℓ) , then \mathcal{K} also satisfies the $(\alpha, 0)$ -good neighborhood property for the clustering problem (S, ℓ) .*

Fact 2 *If the similarity function \mathcal{K} satisfies the ν -strict separation property for the clustering problem (S, ℓ) , then \mathcal{K} also satisfies the $(0, \nu)$ -good neighborhood property for the clustering problem (S, ℓ) .*

Balkan et. al [2] have shown that if \mathcal{K} satisfies the strict separation property with respect to the target clustering, then as long as the smallest target cluster has size $5\nu n$, one can in polynomial time construct a hierarchy with the guarantee that the ground-truth is ν -close to a pruning of the hierarchy. Unfortunately the algorithm presented in [2] is computationally very expensive: it first generate a large list of $\Omega(n^2)$ candidate clusters and repeatedly runs pairwise tests in order to laminarize these clusters; its running time is a large unspecified polynomial. Our new robust linkage algorithm can be used to get a simpler and much faster algorithm for clustering accurately under the ν -strict separation property. Additionally, our algorithm is much more general as well.

As shown in [2], the $(2, \epsilon)$ BBG-condition for k-median implies the ν -strict separation condition [1], for $\nu = 5\epsilon$. One can show a similar result for the (ν, c, ϵ) -condition for k-median introduced by [3], and so the condition we analyze is strictly more general than these conditions.

As we show below, if the data satisfies the good neighborhood property, then most of the standard linkage based algorithms will fail. The contribution of our paper is to develop a robust, linkage based algorithm that will succeed under these natural conditions.

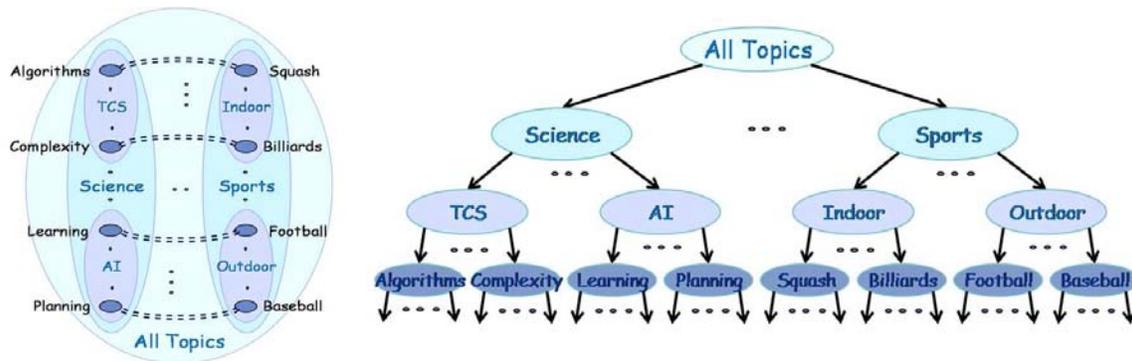


Figure 2: Same as Figure 1 except let us match each point in Algorithms with a point in Squash, each point in Complexity with a point in Billiards, each point in Learning with a point in Football, and each point in Planning with a point in region Baseball. Define the similarity measure to be the same as in Figure 1 except that we let $\mathcal{K}(x, y) = 1$ if x and y are matched. Note that for $\alpha = 1/n$ the similarity function satisfies the α -good neighborhood property with respect to any of the prunings of the tree above. However, single linkage, average linkage, and complete linkage would initially link the matched pairs and produce clusters with very high error with respect to any such clustering.

2.2 Standard linkage based algorithms are not robust

We can show an example where simple linkage based algorithm would perform very badly, but where our algorithm would work well. In particular, if we slightly modify the example in Figure 1, by adding a little bit of noise, to form links of high similarity between points in different inner blobs, we can show that many of the classic linkage based algorithms will perform poorly². See Figure 2 for a precise description of the similarity measure.

In particular, the single linkage algorithm, the average linkage algorithm, and the complete linkage algorithm, would in the first $n/2$ stages merge the matched pairs of points. From that moment on, no matter how they perform, none of the natural and desired clusterings will be even $1/2$ close to any of the prunings of the hierarchy produced. Notice however, that the similarity \mathcal{K} satisfies α -good neighborhood property with respect to any of the desired clusterings (for $\alpha = 1/n$), and that our algorithm will be successful on this instance. The ν -strict separation is not satisfied in this example either, for any constant ν .

3 Robust Hierarchical Clustering

In this section we describe an algorithm that we prove is successful if the data satisfies the good neighborhood property. This procedure has two phases: first, it uses somewhat more global information for creating an interesting starting point for a linkage procedure – a set of not too small, but also not too large blobs that are mostly “pure”. In a second phase, it runs robust linkage procedure on this set of blobs. Both steps have to be done with care and we will describe in detail in the following sections both steps of our algorithm. In particular, in section 3.1 we describe the procedure for generating a set of interesting blobs and in section 3.2 we describe the linkage procedure.

We start with a useful definition.

Definition 1 For $A \subseteq S$, $B \subseteq S$ we define $\mathcal{K}_{\text{median}}(A, B) = \text{median}\{\mathcal{K}(x, x'); x \in A, x' \in B\}$ and we call this the median similarity of A to B .

For simplicity we denote $\mathcal{K}_{\text{median}}(\{x\}, B)$ as $\mathcal{K}_{\text{median}}(x, B)$.

Notation: For the rest of this section we assume that the similarity function \mathcal{K} satisfies the (α, ν) -good neighborhood property for the clustering problem (S, ℓ) . Let $S' \subseteq S$ be the set of $(1 - \nu)n$

¹Note that we assume that for any given point we have a canonical order for its neighbors, and so the set of t nearest neighbors of a given point is always well defined.

²Since, usually, the similarity function between pairs of objects is constructed based on heuristics, this can easily happen; for example we could have a similarity measure that puts a lot of weight on features such as date or names, and so we could easily have a document about Learning being more similar to a document about Football than to other documents about Learning.

Algorithm 1 Robust Agglomerative Hierarchical Clustering

Input: Similarity function \mathcal{K} , set of points S , $\nu > 0$, $\alpha > 0$.

1. Run Algorithm 2 with parameters ν , α to generate an interesting list L of blobs that partitions the whole set S .
2. Run the linkage Algorithm 3 on these blobs to get the tree T .

Output: Tree T on subsets of S .

points such that \mathcal{K} satisfies α -good neighborhood property with respect to S' . We call the points in S' good points and the points in $S \setminus S'$ bad points. Let $G_i = C_i \cap S'$ be the good set of label i . Let $G = \cup G_i$ the whole set of good points; so $G = S'$. Clearly $|G| \geq n - \nu n$. Denote by \mathcal{C}_G the restriction of the target clustering to the set G .

Note that the following is a useful consequence of the definition.

Claim 2 *Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (S, ℓ) . As long as t is smaller than n_{C_i} for any good point $x \in C_i$ all but at most $(\nu + \alpha)n$ out of its t nearest neighbors lie in its good set $C_i(x) \cap G$.*

3.1 Generating an interesting starting point

In this section we describe our procedure for generating a set of interesting blobs, i.e., a set of not too small, but also not too large blobs that are almost pure.

Algorithm 2 Generate interesting blobs

Input: similarity function \mathcal{K} , set of points S , $\nu > 0$, $\alpha > 0$.

Let the initial threshold $t = 6(\nu + \alpha)n + 1$. Let L be empty. Let $A_S = S$.

- Step 1** Construct the graph F_t where we connect points x and y in A_S if they share at least $t - 2(\nu + \alpha)n$ points in common out of their t nearest neighbors with respect to the whole set of points S .
- Step 2** Construct the graph H_t by connecting points x and y if they share at least $3(\nu + \alpha)n$ neighbors in the graph F_t .
- Step 3** (i) Add to L all the components C of H_t with $|C| \geq 3(\nu + \alpha)n$ and remove from A_S all the points in all these components.
(ii) For all points x in A_S check if $(\nu + \alpha)n$ out of their $5(\nu + \alpha)n$ nearest neighbors are in L . If so, then assign point x to any of the blobs in L of highest median. Remove the points in all these components from A_S .
- Step 4** While $|A_S| \geq 3(\nu + \alpha)n$ and $t < n$, increase the critical threshold and go to Step 1.
- Step 5** Assign all points x that do not belong to any of the blobs in L arbitrarily to one of the blobs.

Output: A list of blobs which form a partition of S .

We can show the following:

Theorem 3 *Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (S, ℓ) . So long as the smallest target cluster has size greater than $9(\nu + \alpha)n$, then we can use Algorithm 2 to create a list L of blobs each of size at least $3(\nu + \alpha)n$ such that:*

- The blobs in L form a partition of S .
- Each blob in the list L contains good points from only one good set; i.e., for any $C \in L$, $C \cap G \subseteq G_i$ for some $i \leq k$.

Proof: In the following we denote by n_{C_i} the number of points in the target cluster i . Without loss of generality assume that $n_{C_1} \leq n_{C_2} \leq \dots \leq n_{C_k}$. We will show by inductions on $i \leq k$ that:

- (a) For any $t \leq n_{C_i}$, any blob in the list L only contains good points from a single good set G_i ; all blobs have size at least $3(\nu + \alpha)n$.
- (b) At the beginning of the iteration $t = n_{C_i} + 1$, any good point $x \in C_j \cap G$, $j \in \{1, 2, \dots, i\}$ has already been assigned to a blob in the list L that contains points only from $C_j \cap G$ and has more good points than bad points.

These two claims clearly imply that each blob in the list we output contains good points from only one good set. Moreover at $t = n_{C_k}$ all good points have been assigned to one of the blobs in L . Since we assign the remaining points x that do not belong to any of the blobs in L (these can only be bad points) arbitrarily to one of the blobs, we also get that the blobs in L form a partition of S , as desired.

Claims (a) and (b) are clearly both true initially. We show now that as long as $t \leq n_{C_1}$, the graphs F_t and H_t have the following properties:

- (1) No good point in cluster i is connected in F_t to a good point in a different cluster j , for $i, j \geq 1$, $i \neq j$. Since \mathcal{K} satisfies the (α, ν) -good neighborhood property for the clustering problem (S, ℓ) , by Claim 2, we know that as long as t is smaller than n_{C_i} for any good point $x \in C_i$ all but at most $(\nu + \alpha)n$ out of its t nearest neighbors lie in its good set, i.e., $C_i(x) \cap G$; similarly, as long as t is smaller than n_{C_j} for any good point $y \in C_j$, all but at most $(\nu + \alpha)n$ out of its t nearest neighbors lie in its good set, i.e., $|C_j(x) \cap G|$; so it cannot be the case that for $6(\nu + \alpha)n \leq t \leq n_{C_1}$ two good points in two different clusters $i, j \geq 1$ share $t - 2(\nu + \alpha)n$ points in common out of their t nearest neighbors.
- (2) No bad point is connected in F_t to both a good point in cluster i and a good point in different cluster j , for $i, j \geq 1$, $i \neq j$. This again follows from the fact that since $t \leq n_{C_i}$ for all i , for any good point x all but at most $(\nu + \alpha)n$ out of its t nearest neighbors lie in its good set $C_i(x) \cap G$ (by Claim 2); so for a bad point z to share $t - 2(\nu + \alpha)n$ points out of its t nearest neighbors in common with the t nearest neighbors of a good point x in G_i it must be the case that z has $t - 3(\nu + \alpha)n$ points out of its t nearest neighbors in G_i ; but that means that there cannot be other good point y in G_j , where $j \neq i$ such that z and y share $t - 2(\nu + \alpha)n$ points among their t nearest neighbors, because we would need to have that $t - 3(\nu + \alpha)n$ of points out of z 's t nearest neighbors lie in G_i and $t - 3(\nu + \alpha)n$ of points out of z 's t nearest neighbors in G_j ; but this cannot happen if $t > 6(\nu + \alpha)n$ since $2(t - 3(\nu + \alpha)n) > t$ for $t > 6(\nu + \alpha)n$.
- (3) All the components of H_t of size at least $3(\nu + \alpha)n$ will only contain good points from one cluster. Since in F_t bad points can only connect to one good set, we get that no two good points in the different clusters connect in H_t .

We can use (1), (2), and (3) to argue that as long as $t \leq n_{C_1}$, each blob in L contains good points from at most one target cluster. This is true at the beginning and by (3), for any $t \leq n_{C_1}$, anytime we insert a whole new blob in L in Step 3(i), that blob must contain good points from at most one target cluster. We now argue that this property is never violated as we assign points to blobs already in L based on the median test in Step 3(ii). Note that at all time steps all the blobs in L have size at least $3(\nu + \alpha)n$. Assume that a good point x has more than $(\nu + \alpha)n$ out of its $5(\nu + \alpha)n$ nearest neighbors in S in the list L . By Lemma 5, there must exist a blob in L that contains only good points from $C(x)$. By Lemma 4, if we assign x based on the median test in Step 3(ii), then we will add x to a blob containing good points only from $C(x)$, and so we maintain the invariant that each blob in L contains good points from at most one target cluster.

We now show that at the beginning of the iteration $t = n_{C_1} + 1$, all the good points in C_1 have already been assigned to a blob in the list L that only contains good points from $C_1 \cap G$. There are a few cases. First, if prior to $t = n_{C_1}$ we did not yet extract in the list L a blob with good points from C_1 , then it must be the case that all good points in C_1 connect to each other in the graph F_t ; so there will be a component of H_t that will contain all good points from C_1 and potentially bad points, but no good points from another target cluster; moreover this $|C_1| \geq 9(\nu + \alpha)n$, this component will be output in Step 3(i). Second, if prior to $t = n_{C_1}$ we did extract some, but still, more than $3(\nu + \alpha)n$ points from the good set G_1 do not belong to blobs in the list L , then more than $3(\nu + \alpha)n$ of good points will connect to each other in F_t , and then in H_t , so we will add one blob to L containing these good points (plus at most νn bad points). Finally, it could be that by the time we reach $t = n_{C_1}$ all but $l < 3(\nu + \alpha)n$ good points in C_1 have been assigned to a blob in the list L that has good points only from C_1 . Since $|C_1| \geq 9(\nu + \alpha)n$ we must have assigned at least $9(\nu + \alpha)n - 3(\nu + \alpha)n - \nu n \geq 5(\nu + \alpha)n$ good points from C_1 to the list L . This together with

the (α, ν) -good neighborhood property implies that the good points in C_1 that do not belong to the list L yet, must have $(\nu + \alpha)n$ out of their $5(\nu + \alpha)n$ nearest neighbors in S in the list L (at most ν out of the $5(\nu + \alpha)n$ nearest neighbors can be bad points, at most αn can be good points from a different cluster, and at most $3(\nu + \alpha)n$ can be good points in C_1 that do not yet belong to L). So we will assign these points to blobs in L based on the median test in Step 3(ii). By Lemma 4, when we assign them based on the median test in Step 3(ii), we will add them to a blob containing good points from C_1 and no good points from other cluster C_j , as desired.

We then iterate the argument on the remaining set A_S . The key point is that for $t \geq n_i$, $i > 1$, once we start analyzing good points in $C_{n_{i+1}}$ we have that *all* the good points in C_{n_i} , $C_{n_{i-1}}$, ..., C_{n_1} have already been assigned to blobs in L . ■

We prove below two useful lemmas used in the above proof.

Lemma 4 *Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (S, ℓ) . Assume that L is a list of disjoint clusters each of size at least $3(\nu + \alpha)n$. Assume also that each cluster in L intersects at most a good set; i.e., for any C in L , we have $C \cap G \subseteq G_i$ for some i . Consider $x \in G$ such that there exist C in L with $C \cap G \subseteq C(x) \cap G$. Let \tilde{C} be the blob in L of highest median similarity to x . Then $\tilde{C} \cap G \subseteq C(x) \cap G$.*

Proof: Let us fix a good point x . Let C' and C'' be such that $C' \cap G \subseteq C(x) \cap G$ and $C'' \cap G \subseteq C_i \cap G$, for $C_i \neq C(x)$. Since \mathcal{K} is a symmetric similarity function satisfying the (α, ν) -good neighborhood property, by Claim 2, we have that x can be more similar to at most $\nu n + \alpha n$ points in C'' than with any point in $C' \cap G$. Since $|C'| \geq 3(\nu + \alpha)n$ and $|C''| \geq 3(\nu + \alpha)n$ we get that

$$\mathcal{K}_{\text{median}}(x, C') \geq \mathcal{K}_{\text{median}}(x, C'').$$

This then implies that the blob \tilde{C} in L of highest median similarity to x must satisfy $\tilde{C} \cap G \subseteq C(x) \cap G$, as desired. ■

Lemma 5 *Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (S, ℓ) . Assume that L is a list of clusters each of size at least $3(\nu + \alpha)n$. Assume also that each cluster in L intersects at most a good set; i.e., for any C in L , we have $C \cap G \subseteq G_i$ for some i . Consider $x \in G$ such that there is no cluster C in L with $C \cap G \subseteq C(x) \cap G$. Then at most $(\alpha + \nu)n$ of its t nearest neighbors for any $t \leq n_{C(x)}$ can be in L and all the rest are outside.*

Proof: Since \mathcal{K} satisfies the (α, ν) -good neighborhood property, by Claim 2, for all $x \in G$, for all $t \leq n_{C(x)}$ at most $(\alpha + \nu)n$ of its t nearest neighbors are not from $C(x)$. Consider $x \in G$ such that there is no cluster C in L with $C \cap G \subseteq C(x) \cap G$. So, the list L contains no good points from C , which implies that at most $(\alpha + \nu)n$ of its t nearest neighbors for any $t \leq n_{C(x)}$ can be in L . ■

3.2 A robust linkage procedure

Our linkage procedure is given by Algorithm 3.

Algorithm 3 A robust linkage procedure

Input: A list L of blobs; similarity function \mathcal{K} on pairs of points.

- Repeat till only one cluster remains in L :
 - (a) Find clusters C, C' in the current list which maximize score(C, C')
 - (b) Remove C and C' from L merge them into a single cluster and add that cluster to L .
- Let T be the tree with single elements as leaves and internal nodes corresponding to all the merges performed.

Output: Tree T on subsets of S .

We describe in the following the notion of similarity between pairs of blobs used in Algorithm 3.

Definition 6 *Let $L = \{A_1, \dots, A_l\}$ be a list of disjoint subsets of S . For each i , for each point x in A_i we compute $\mathcal{K}_{\text{median}}(\{x\}, A_j)$, $j \neq i$, sort them in increasing order, and define rank(x, A_j) as the rank of A_j in the order induced by x . We define*

$$\text{rank}(A_i, A_j) = \text{median}_{x \in A_i} [\text{rank}(x, A_j)].$$

For example, if A_{j_1} is the subset of highest median similarity to x out of all A_j , $j \neq i$, then $\text{rank}(x, A_{j_1}) = l$. Similarly, if A_{j_2} is the subset of smallest median similarity to x out of all A_j , $j \neq i$, then $\text{rank}(x, A_{j_2}) = 1$.

Definition 7 Let $L = \{A_1, \dots, A_l\}$ be a list of disjoint subsets of S . We define the score between A_i and A_j as

$$\text{score}(A_i, A_j) = \min[\text{rank}(A_i, A_j), \text{rank}(A_j, A_i)].$$

Note that while the $\text{rank}(\cdot, \cdot)$ might be asymmetric, $\text{score}(\cdot, \cdot)$ is designed to be symmetric. We now present a useful lemma.

Lemma 8 Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (S, ℓ) . Let L be a list of disjoint clusters, all of size at least $2(\alpha + \nu)n$. Assume that $B \cap G \subseteq G_i$, $B' \cap G \subseteq G_i$, and $(B'' \cap G) \cap G_i = \emptyset$. Then we have both

$$\text{score}(B, B') < \text{score}(B, B'') \quad \text{and} \quad \text{score}(B, B') < \text{score}(B', B'').$$

Proof: Let x be a good point. The (α, ν) -good neighborhood property implies that there exists c_x such that at most αn points $z \in G$, $z \notin C(x)$ can have similarity $K(x, z)$ greater or equal to c_x and at most αn points $y \in G \cap C(x)$ can have similarity $K(x, z)$ strictly smaller than c_x . Since each of the blobs has size at least $2(\alpha + \nu)n$ and since each blob contains at most νn bad points, we get that for all blobs B' and B'' such that $B' \cap G \subseteq C(x) \cap G$ and $(B'' \cap G) \cap (C(x) \cap G) = \emptyset$ we have

$$\mathcal{K}_{\text{median}}(\{x\}, B') > \mathcal{K}_{\text{median}}(\{x\}, B'').$$

So a good point x will rank blobs B' s.t. $B' \cap G \subseteq C(x) \cap G$ later than blobs B'' such that $(B'' \cap G) \cap (C(x) \cap G) = \emptyset$ in the order it induces. Assume that there are exactly r blobs B in L such that $(B \cap G) \cap (C(x) \cap G) = \emptyset$. Since there are at most νn bad points and each of the blobs has size at least $2(\alpha + \nu)n$, we obtain that for all B, B' in L such that $B \cap G \subseteq C_i \cap G$ and $B' \cap G \subseteq C_i \cap G$, and for all B'' in L with $(B'' \cap G) \cap (C_i \cap G) = \emptyset$ we have both

$$\text{rank}(B, B') > r \geq \text{rank}(B, B'') \quad \text{and} \quad \text{rank}(B', B) \geq r \geq \text{rank}(B', B'').$$

This then implies that

$$\text{score}(B, B') = \text{score}(B', B) = \min[\text{rank}(B, B'), \text{rank}(B', B)] > r.$$

Similarly,

$$\text{score}(B, B'') = \text{score}(B'', B) = \min[\text{rank}(B, B''), \text{rank}(B'', B)] > r.$$

Finally, we have

$$\text{score}(B', B'') = \text{score}(B'', B') = \min[\text{rank}(B', B''), \text{rank}(B'', B')] \leq r.$$

These imply:

$$\text{score}(B, B') > \text{score}(B, B'') \quad \text{and} \quad \text{score}(B, B') > \text{score}(B', B''),$$

as desired. ■

We now show that is the similarity function we have satisfies the good neighborhood property, given a good starting point, Algorithm 3 will be successful in outputting a good hierarchy.

Theorem 9 Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (S, ℓ) . Assume that L is a list of clusters each of size at least $3(\nu + \alpha)n$ that partition the entire set of points. Assume also that each cluster in L intersects at most a good set; i.e., for any C in L , we have $C \cap G \subseteq G_i$ for some i . Then Algorithm 3 constructs a binary tree such that the ground-truth clustering is ν -close to a pruning of this tree.

Proof: First note that at each moment the list L of clusters is a partition of the whole dataset and that all clusters in L have size at least $3(\nu + \alpha)n$. We prove by induction that at each time step the list of clusters restricted to G is laminar w.r.t. \mathcal{C}_G .

In particular, assume that our current list of clusters restricted to G is laminar with respect to \mathcal{C}_G (which is true at the start). This implies that for each cluster C in our current clustering and each C_r in the ground truth, we have either

$$C \cap G \subseteq G(C_r) \quad \text{or} \quad G(C_r) \subseteq C \cap G \quad \text{or} \quad (C \cap G) \cap G(C_r) = \emptyset.$$

Now, consider a merge of two clusters C and C' . The only way that laminarity could fail to be satisfied after the merge is if for one of the two clusters, say, C' , we have that $C' \cap G$ is strictly contained inside $C_{r'} \cap G$, for some ground-truth cluster $C_{r'}$ (so, $(C_{r'} \cap G) \setminus (C' \cap G) \neq \emptyset$, $(C' \cap G) \subset C_{r'}$) and yet $C \cap G$ is disjoint from $C_{r'} \cap G$. But there must exist C'' in the list L such that

$$(C'' \cap G) \subset C_{r'} \setminus (C' \cap G), \quad |C''| \geq 3(\nu + \alpha)n.$$

By Lemma 8 we know that

$$\text{score}(C', C'') > \text{score}(C', C).$$

However, this contradicts the specification of the algorithm, since by definition it merges the pair C, C' such that $\text{score}(C', C)$ is greatest. ■

3.3 The Main Result

Our main result is the following:

Theorem 10 *Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (S, ℓ) . As long as the smallest target cluster has size greater than $9(\nu + \alpha)n$, then we can use Algorithm 1 in order to produce a tree such that the ground-truth clustering is ν -close to a pruning of this tree in $O(n^{\omega+1})$ time, where $O(n^\omega)$ is the state of the art for matrix multiplication.*

Proof: The correctness follows immediately from Theorems 3 and 9. For a proof of the running time see the full version of the paper [4]. ■

3.4 The Inductive Setting

In this section we consider an *inductive* model in which S is merely a small random subset of points from a much larger abstract instance space X . Based on such a sample, our algorithm outputs a hierarchy over the sample, which also *implicitly* represents a hierarchy of the whole space which is evaluated with respect to the underlying distribution. Let us assume for simplicity that X is finite and that the underlying distribution is uniform over X .

Our goal is to design an algorithm that based on the sample produces a tree of small error with respect to the whole distribution. Formally, we assume that each node in the tree derived over the sample S induces a cluster (a subset of X) which is implicitly represented as a function $f : X \rightarrow \{0, 1\}$. For a fixed tree T and a point x , we define $T(x)$ as the subset of nodes in T that contain x (the subset of nodes $f \in T$ with $f(x) = 1$). We say that a tree T has error at most ϵ if $T(X)$ has a pruning $f_1, \dots, f_{k'}$ of error at most ϵ .

Algorithm 4 Inductive Robust Agglomerative Hierarchical Clustering

Input: Similarity function \mathcal{K} , parameters $\alpha, \nu, k \in \mathbb{Z}^+, \delta; n = n(\alpha, \nu, k, \delta)$;

- Pick a set $S = \{x_1, \dots, x_n\}$ of n random examples from X .
 - Run Algorithm 1 with parameters $2\alpha, 2\nu$ on the set S and obtain a tree T on the subsets of S . Let Q be the set of leaves of this tree.
 - Associate each node u in T a function f_u (which induces a cluster) specified as follows:
Consider $x \in X$, and let $q(x) \in Q$ be the leaf given by $\text{argmax}_{q \in Q} \mathcal{K}_{\text{median}}(x, q)$; if u appears on the path from $q(x)$ to the root, then set $f_u(x) = 1$, otherwise set $f_u(x) = 0$.
 - Output the tree T .
-

Let $N = |X|$. For the rest of this section we assume that the similarity function \mathcal{K} satisfies the (α, ν) -good neighborhood property for the clustering problem (X, ℓ) . Let $S' \subseteq S$ be the set of $(1 - \nu)N$ points such that \mathcal{K} satisfies α -good neighborhood property with respect to S' . We call the points in S' good points and the points in $S \setminus S'$ bad points. Let $G_i = C_i \cap S'$ be the good set of label i . Let $G = \cup G_i$ the whole set of good points; so $G = S'$.

Our main result in this section is the following:

Theorem 11 *Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (X, ℓ) . As long as the smallest target cluster has size greater than $18(\nu + \alpha)N$, then using Algorithm 4 with parameters α, ν, k, δ , and $n = \Theta\left(\frac{1}{\min(\alpha, \nu)} \ln \frac{k}{\delta \cdot \min(\alpha, \nu)}\right)$, we can produce a tree with the property that the ground-truth is $\nu + \delta$ -close to a pruning of this tree with probability $1 - \delta$. Moreover, the size of this tree is $O\left(\frac{1}{\min(\alpha, \nu)} \ln \frac{k}{\delta \cdot \min(\alpha, \nu)}\right)$.*

Proof: Note that n is large enough so that with probability at least $1 - \delta/2$ we have that S contains at most $2\nu n$ bad points and \mathcal{K} satisfies the $(2\alpha, 2\nu)$ -good neighborhood property with respect to the clustering induced over the sample (by Lemma 12) and each target cluster has at least $9(\nu + \alpha)n$ points in the sample (by Chernoff bounds).

Assume below that this happens. So, by Theorem 10 we get that the tree T induced over the sample has error at most 2ν over the sample. Let L be the list of leaves of T . By Theorem 3, we know that L forms a partition of S and that each element of L has size at least $6(\nu + \alpha)n$ and it contains good points from only one good set i.e., for any $C \in L$, $C \cap G \subseteq G_i$ for some $i \leq k$. Let us fix a good point x . By Lemma 13, with probability at least $1 - \delta^2/2$ at most $2\alpha n$ of the $n_{\tilde{C}_G(x)}$ nearest points to x from $G \cap S$ can be outside $C(x) \cap G$, where $n_{\tilde{C}_G(x)}$ is $|C(x) \cap G \cap S|$. We can show that \tilde{C} , the blob in L of highest median similarity to x , satisfies $\tilde{C} \cap G \subseteq C(x) \cap G$. To see this, let C' and C'' in L be such that $C' \cap G \subseteq C(x) \cap G$ and $C'' \cap G \subseteq C_i \cap G$, for $C_i \neq C(x)$. By the above facts we know that x can be more similar to at most $2\nu n + 2\alpha n$ points in C'' than with any point in $C' \cap G$. Since $|C'| \geq 6(\nu + \alpha)n$ and $|C''| \geq 6(\nu + \alpha)n$ we get that

$$\mathcal{K}_{\text{median}}(x, C') \geq \mathcal{K}_{\text{median}}(x, C'').$$

This then implies that the blob \tilde{C} in L of highest median similarity to x must satisfy $\tilde{C} \cap G \subseteq C(x) \cap G$, as desired. So, for any given point x , with probability $1 - \delta^2/2$ over the draw of the random sample, the leaf in T of highest median similarity to x has the property that all its good points are from $C(x)$. Since this is true for any x , by Markov inequality, we get that with probability $1 - \delta/2$ a $1 - \delta$ fraction of the good points connect to a leaf that contain good points from their own cluster only.

Adding back the $\delta/2$ chance of failure due to either \mathcal{K} not satisfying the $(2\alpha, 2\nu)$ -good neighborhood property or having more than $2\nu n$ bad points in S , we get that with probability $1 - \delta$ the error rate of the hierarchy implied by T over the whole set X is at most $\nu + \delta$. \blacksquare

Lemma 12 *Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (X, ℓ) . If we draw a set S of $n = \Theta\left(\frac{1}{\min(\alpha, \nu)} \ln \frac{1}{\delta \cdot \min(\alpha, \nu)}\right)$, then with probability $1 - \delta$, S contains at most $2\nu n$ bad points and the similarity function \mathcal{K} satisfies the $(2\alpha, 2\nu)$ -good neighborhood property with respect to the target clustering restricted to the sample S .*

Proof: Since $n \geq \frac{3}{\nu} \ln \frac{2}{\delta}$, by Chernoff bounds, we have that with probability $1 - \delta/2$ at most $2\nu n$ bad points fall into the sample. Let us fix a good point x in S and let us denote by $n_{\tilde{C}_G(x)}$ the number of points in $C(x) \cap G \cap S$. By Lemma 13 we have that for $n = \Theta\left(\frac{1}{\alpha} \ln \frac{n}{\delta}\right)$, with probability at least $1 - \delta/(2n)$ (over the draw of the other points in the sample) we have that all but $2\alpha n$ of the $n_{\tilde{C}_G(x)}$ nearest neighbors of x from $S \cap G$ are points from the set $C(x) \cap G$. By union bound over all points x in S we have that simultaneously for all good points x in S , all but $2\alpha n$ of their $n_{\tilde{C}_G(x)}$ nearest neighbors in $S \cap G$ come from $C(x) \cap G$.

These together imply that if $n = \Theta\left(\frac{1}{\min(\alpha, \nu)} \ln \frac{n}{\delta}\right)$, then with probability $1 - \delta$ at most $2\nu n$ bad points fall into the sample and the similarity function \mathcal{K} satisfies the $(2\alpha, 2\nu)$ -good neighborhood property with respect to the target clustering restricted to the sample S . Using the inequality

$$\ln x \leq \alpha x - \ln \alpha - 1$$

for $\alpha, x > 0$, we then get the desired result. \blacksquare

Lemma 13 *Let \mathcal{K} be a symmetric similarity function satisfying the (α, ν) -good neighborhood property for the clustering problem (X, ℓ) . Consider $x \in G$. If we draw a set S of $n = \Theta\left(\frac{1}{\alpha} \ln \frac{1}{\delta}\right)$ random points from X , then with probability at most $1 - \delta$ we have that at most $2\alpha n$ of the $n_{\tilde{C}_G(x)}$ nearest points to x from $G \cap S$ can be outside $C(x) \cap G$, where $n_{\tilde{C}_G(x)}$ is $|C(x) \cap G \cap S|$.*

Proof: Let $C_G(x)$ denote $C(x) \cap G$ and let

$$n_{C_G(x)} = |C(x) \cap G|.$$

Let us define $NN(x)$ to be the nearest $n_{C_G(x)}$ points to x in G . Since \mathcal{K} satisfies the (α, ν) -good neighborhood property for the clustering problem (X, ℓ) we have:

$$Pr_{z \sim X}[z \in NN(x) \setminus C_G(x)] \leq \alpha.$$

Since $NN(x)$ and $C_G(x)$ have the same size, this is equivalent to the statement:

$$Pr_{z \sim X}[z \in C_G(x) \setminus NN(x)] \leq \alpha.$$

So, by Chernoff bounds applied to both of the above, with probability at most $1 - \delta$ we have that at $2\alpha n$ points are in $(NN(x) \setminus C_G(x)) \cap S$ and at most $2\alpha n$ points are in $(C_G(x) \setminus NN(x)) \cap S$.

We now argue that at most $2\alpha n$ of the $n_{\tilde{C}_G(x)}$ nearest points to x in $G \cap S$ can be outside $C(x) \cap G$, where $n_{\tilde{C}_G(x)} = |C(x) \cap G \cap S|$. Let

$$n_1 = |(NN(x) \setminus C_G(x)) \cap S|,$$

$$n_2 = |(C_G(x) \setminus NN(x)) \cap S|,$$

$$n_3 = |(C_G(x) \cap NN(x)) \cap S|.$$

By construction, we have

$$n_{\tilde{C}_G(x)} = n_2 + n_3,$$

and we are given that $n_1, n_2 \leq 2\alpha n$. We now distinguish two cases.

The first case is $n_1 \geq n_2$. In this case we have

$$n_1 + n_3 \geq n_2 + n_3 = n_{\tilde{C}_G(x)}.$$

This implies that the nearest $n_{\tilde{C}_G(x)}$ points to x in $G \cap S$ all lie inside $NN(x)$, since by definition all points inside $NN(x)$ are closer to x than any point in G outside $NN(x)$. Since we are given that at most $n_1 \leq 2\alpha n$ of them can be outside $C_G(x)$, we get that at most $2\alpha n$ of the $n_{\tilde{C}_G(x)}$ nearest neighbors of x are not from $C_G(x)$, as desired.

The second case is $n_1 \leq n_2$. This implies that the nearest $n_{\tilde{C}_G(x)}$ good points to x in the sample include *all* the points in $NN(x)$ in the sample, plus possibly some others too. But this implies in particular that it includes all the n_3 points in $C_G(x) \cap NN(x)$ in the sample. So, it can include at most

$$n_{\tilde{C}_G(x)} - n_3 \leq 2\alpha \cdot n$$

points not in $C_G(x) \cap NN(x)$, and even if all those are not in $C_G(x)$, it is still $\leq 2\alpha n$; so at most $2\alpha n$ of the $n_{\tilde{C}_G(x)}$ nearest neighbors of x are not from $C_G(x)$, as desired. ■

Note: Note that if we are willing to lose a bit in the accuracy the analysis in this section allows us to speed up the algorithm in Theorem 10.

4 Discussion and Open Questions

In this paper we propose and analyze a new robust algorithm for hierarchical clustering. We show that our algorithm can be used to cluster accurately in interesting cases where traditional agglomerative algorithms fail. In particular, we show that our algorithm is provably correct if the data satisfies a natural good neighborhood property, a relaxation of the strict separation property that allows for substantial degrees of noise.³ We also show how to adapt our algorithm to the inductive setting where our given data is only a small random sample of the entire data set.

The running time of our algorithm is currently $O(n^{\omega+1})$, where $O(n^\omega)$ is the state of the art for matrix multiplication and n is either the size of the dataset or the size of the sample in the inductive case. It would be interesting to develop even faster algorithms achieving the same accuracy guarantees.

It would also be interesting to see if our algorithmic approach can be shown to work for other natural properties. For example, it would be particularly interesting to analyze a noisy versions of the max stability property in [2] which was shown to be a necessary and sufficient condition for single linkage to succeed, or of the average stability property which was shown to be a sufficient condition for average linkage to succeed.

³As mentioned in the introduction, most of the traditional agglomerative algorithms would succeed under the strict separation property with no noise, but even tiny amount of noise would cause them to fail badly.

Acknowledgments:

We thank Avrim Blum for numerous useful discussions. This work was supported in part by NSF grant CCF-0953192, by ONR grant N00014-09-1-0751, and AFOSR grant FA9550-09-1-0538.

References

- [1] M. F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *ACM-SIAM Symposium on Discrete Algorithms*, 2009.
- [2] M. F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *40th ACM Symposium on Theory of Computing*, 2008.
- [3] M. F. Balcan, H. Roeglin, and S. Teng. Agnostic clustering. In *The 20th International Conference on Algorithmic Learning Theory*, 2009.
- [4] M.F. Balcan and P. Gupta. Robust hierarchical clustering. Technical Report GT-CS-10-06, 2010.
- [5] D. Bryant and V. Berry. A structured family of clustering and tree construction methods. *Advances in Applied Mathematics*, 27(4):705–732, 2001.
- [6] D. Cheng, R. Kannan, S. Vempala, and G. Wang. A divide-and-merge methodology for clustering. *ACM Trans. Database Syst.*, 31(4):1499–1525, 2006.
- [7] S. Dasgupta and P. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555 – 569, 2005.
- [8] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2000.
- [9] S. Gollapudi, R. Kumar, and D. Sivakumar. Programmable clustering. In *Symposium on Principles of Database Systems*, pages 348 – 354, 2006.
- [10] K. Heller. *Efficient Bayesian Methods for Clustering*. PhD thesis, 2008.
- [11] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [12] A. K. Jain, M. N. Murthy, and P. J. Flynn. Data clustering: A review. In *ACM Computing Reviews*, 1999.
- [13] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.
- [14] M. Narasimhan, N. Jojic, and J. Bilmes. Q-clustering. In *Neural Information Processing Systems*, 2005.
- [15] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101, 2006.

Theoretical Justification of Popular Link Prediction Heuristics

Purnamrita Sarkar
Carnegie Mellon University
psarkar@cs.cmu.edu

Deepayan Chakrabarti
Yahoo! Research
deepay@yahoo-inc.com

Andrew W. Moore
Google, Pittsburgh
awm@google.com

Abstract

There are common intuitions about how social graphs are generated (for example, it is common to talk informally about *nearby* nodes sharing a link). There are also common heuristics for predicting whether two currently unlinked nodes in a graph should be linked (e.g. for suggesting friends in an online social network or movies to customers in a recommendation network). This paper provides what we believe to be the first formal connection between these intuitions and these heuristics. We look at a familiar class of graph generation models in which nodes are associated with locations in a latent metric space and connections are more likely between closer nodes. We also look at popular link-prediction heuristics such as number-of-common-neighbors and its weighted variants (Adamic & Adar, 2003) which have proved successful in predicting missing links, but are not direct derivatives of latent space graph models. We provide theoretical justifications for the success of some measures as compared to others, as reported in previous empirical studies. In particular we present a sequence of formal results that show bounds related to the role that a node's degree plays in its usefulness for link prediction, the relative importance of short paths versus long paths, and the effects of increasing non-determinism in the link generation process on link prediction quality. Our results can be generalized to any model as long as the latent space assumption holds.

1 Introduction

Link prediction is a key problem in graph mining. It underlies recommendation systems (e.g., movie recommendations in Netflix, music recommendation engines like `last.fm`), friend-suggestions in social networks, market analysis, and so on. As such, it has attracted a lot of attention in recent years, and several heuristics for link prediction have been proposed (Adamic & Adar, 2003). In-depth empirical studies comparing these heuristics have also been conducted (Liben-Nowell & Kleinberg, 2003) and (Brand, 2005), and two observations are made consistently: (1) a simple heuristic, viz., predicting links between pairs of nodes with the most common neighbors, often outperforms more complicated heuristics, (2) a variant of this heuristic that weights common neighbors using a carefully chosen function of their degrees (Adamic & Adar, 2003) performs even better on many graphs and (3) heuristics which use an ensemble of short paths between two nodes (Katz, 1953) often perform better than those which use longer paths. However, there has been little theoretical work on why this should be so. We present, to our knowledge, the first theoretical analysis of link prediction on graphs. We show how various heuristics compare against each other, and under what conditions would one heuristic be expected to outperform another. We are able to provide theoretical justifications for all of the empirical observations mentioned above.

We define the link prediction problem as follows. There is a latent space in which the nodes reside, and links are formed based on the (unknown) distances between nodes in this latent space. Individual differences between nodes can also be modeled with extra parameters. The quality of link prediction now depends on the quality of estimation of distance between points. We show how different estimators provide *bounds* on distance. Clearly, the tighter the bounds, the better we can distinguish between pairs of nodes, and thus the better the quality of link prediction.

While any latent space model can be used, we extend a model by (Raftery et al., 2002) due to two characteristics: (1) it is simple to state and analyze, (2) yet, it is powerful enough to show all of the effects that affect estimation, such as node degree, lengths of paths, etc. Our results do not assume any degree distribution on the graph; in fact, they depend on very simple properties that should be generalizable to other models as well.

Our primary contributions are as follows:

- We formulate the link prediction problem as a problem of estimating distances between pairs of nodes, where the nodes lie at unknown positions in some latent space and the observed presence or absence of links between nodes provides clues about their distances.

- We show that the number of common neighbors between a pair of nodes gives bounds on the distance between them, with the upper bound on distance decreasing quickly as the count of common neighbors increases. This justifies the popular heuristic of predicting links simply by picking node pairs with the maximum number of common neighbors.
- Empirical studies (Liben-Nowell & Kleinberg, 2003) have shown that another popular heuristic (Adamic & Adar, 2003) that uses a carefully *weighted* count of common neighbors often outperforms the unweighted count. We present theoretical justification for this, and generalize it to other possible weighting schemes that should be similarly useful.
- Finally, another set of heuristics consider longer paths between pairs of nodes, e.g., hitting-time and other measures based on random walks. Our results here are twofold. (1) We show that while the number of long paths can, indeed, provide bounds on distance, these are looser than the bounds obtained if enough short paths (or ideally, common neighbors) exist. Thus, longer paths are more useful if shorter paths are rare or non-existent. (2) We also show that the bounds obtained from long paths can get much tighter given just the knowledge of *existence* of a short path. Thus, even the existence of a single short path can improve bounds obtained from long paths.
- Our results can be applied to any social network model where: nodes are distributed independently in some latent metric space; probability of a link satisfies *homophily*; given the positions links are independent of each other.

This paper is organized as follows. In section 2 we introduce related work and background on link prediction and latent space models. Sections 3 and 4 consist of a formal relationship between popular heuristics like common neighbors and our graph model with same and distinct radii. In section 5, we analyze the implication of paths of length $\ell > 2$. Section 6 shows how to extend the analysis to handle non-determinism in the link generation process. In section 7, we summarize this paper and discuss several implications of our work.

2 Review of Previous Empirical Studies

We will briefly describe the link prediction problem, and the observations from previous empirical studies. Then we will describe the latent space model we use, and conclude with the relation of this model to link prediction.

2.1 Link Prediction

Many real world graph-mining problems can be framed as link prediction. Popular applications include suggesting friends on Facebook, recommending movies (Netflix, MovieLens) or music (last.fm, Pandora) to users. Link prediction on informal office-network has been shown to be useful for suggesting potential collaborators (Raghavan, 2002). Also in intelligence analysis (Schroeder et al., 2003), link-prediction can suggest potential involvement between a group of individuals, who do not have prior records of interaction.

In general popular graph-based proximity measures like personalized pagerank (Jeh & Widom, 2002), hitting and commute times (Aldous & Fill, 2001), Adamic/Adar (Adamic & Adar, 2003) are used for link prediction on graphs. The experimental setup varies from predicting all absent links at once (Liben-Nowell & Kleinberg, 2003) to predicting the best link for a given node (Brand, 2005),(Sarkar & Moore, 2007).

These papers mostly use co-authorship graphs and movie-recommendation networks. We will sketch a few of the observations from earlier empirical evaluation. We would divide heuristics into roughly two parts, simple variants of the number of common neighbors (Adamic/Adar, Jaccard etc.), and measures based on ensemble of paths. Here is a summary of the most interesting observations from (Liben-Nowell & Kleinberg, 2003; Brand, 2005), and (Sarkar & Moore, 2007).

1. The number of common neighbors performs surprisingly well on most data-sets, and in many cases beats more complex measures (Liben-Nowell & Kleinberg, 2003).
2. Adamic/Adar, which is analogous to common neighbors with a skewed weighting scheme mostly outperforms number of common neighbors (Liben-Nowell & Kleinberg, 2003).
3. Shortest path performs consistently poorly (Liben-Nowell & Kleinberg, 2003) and (Brand, 2005). We have also noted this behavior.
4. Ensemble of paths which looks at long paths (hitting and commute times) does not perform very well (Liben-Nowell & Kleinberg, 2003) and (Brand, 2005).
5. Ensemble of paths which down-weights long paths exponentially (e.g. Katz, personalized pagerank) perform better than those which are sensitive to long paths (Liben-Nowell & Kleinberg, 2003) and (Brand, 2005).

While these are interesting results, there has not been any work which theoretically justifies this behavior of different heuristics for link prediction on social networks. In our work, we analyze a simple social network model to justify these empirical results.

2.2 Latent Space Models for Social Network Analysis

Social network analysis has been an active area of research in sociology and statistics for a long time. One important assumption in social networks is the notion of homophily. (McPherson et al., 2001) write in their well-known paper, “*Similarity breeds connection. This principle—the homophily principle—structures network ties of every type, including marriage, friendship, work, advice, support, information transfer, exchange, comembership, and other types of relationship.*” More formally, there is a higher probability of forming a link, if two nodes have similar characteristics. These characteristics can be thought of as different features of a node, i.e. geographic location, college/university, work place, hobbies/interests etc. This notion of *social space* has been examined by (McFarland & Brown, 1973) and (Faust, 1988).

In 2002, (Raftery et al., 2002) introduced a statistical model which explicitly associates every node with locations in a D -dimensional space; links are more likely if the entities are close in latent space. In the original model, the probability of a link between two nodes is defined as a logistic function of their distance. All the pairwise events are independent, conditioned on their latent positions, i.e. distances in the latent space. We alter this model to incorporate radius r in the exponent (for the RHH model $r = 1$). r can be interpreted as the sociability of a node. We name this model- the non-deterministic model (section 6).

$$\text{RHH model: } P(i \sim j | d_{ij}) = \frac{1}{1 + e^{\alpha(d_{ij}-1)}} \quad \text{Our model: } P(i \sim j | d_{ij}) = \frac{1}{1 + e^{\alpha(d_{ij}-r)}}$$

The model has two parameters α and r . Parameter $\alpha \geq 0$ controls the sharpness of the function whereas r determines the threshold. Setting $\alpha = \infty$ in our model, yields a simple deterministic model, on which we build our analysis. It may be noted that given the distances the links are deterministic; but given the links, inferring the distances is an interesting problem. In section 6, we show how this analysis can be carried over to the non-deterministic case with large but finite α . This assumption is reasonable, because low values of α leads to a random graph; $\alpha = 0$ is exactly the $\mathcal{G}_{N,1/2}$ random graph model. Clearly, it is impossible to perform better than a random predictor in a random graph. With distinct radii for each node, the deterministic model can be used for generating both undirected and directed graphs; however for simplicity we use a realistic directed graph model (section 4).

We assume that the nodes are uniformly distributed in a D dimensional Euclidian space. Hence $P(d_{ij} \leq x) = V(x)/V(1)$, where $V(1)$ is the volume of a *unit radius hypersphere*. This uniformity assumption has been made in earlier social network models, e.g. by (Kleinberg, 2000), where the points are assumed to lie on a two dimensional grid. In order to normalize the probabilities, we assume that all points lie inside a *unit volume hypersphere* in D dimensions. The maximum r satisfies $V(r) = V(1)r^D = 1$.

Connection to the Link Prediction Problem. A latent space model is well-fitted for link prediction because, for a given node i , the most likely node it would connect to is the non-neighbor at the smallest distance. The measure of distance comes from the definition of the model, which could be:

1. Undirected graph with identical r . Here distance from node i to j is simply d_{ij} .
2. Directed graph where i connects with j if d_{ij} is smaller than radius of j (or smaller than radius of i) leading to a distance of $d_{ij} - r_j$ (or $d_{ij} - r_i$).

In all these cases, the predicting distances between a pair of nodes is the key. While this can be obtained by maximizing the likelihood of the underlying statistical model, we show that one can obtain high probability bounds on distances from graph based heuristics. In fact we show that the distance to the node picked using a popular heuristic is within a small factor of the *true distance*. This factor quickly goes to zero as N becomes large. Although our analysis uses an extension of the RHH model to actually obtain the bounds on distances, the only property we use is of homophily in a latent metric space, i.e. if two nodes are *close* in some social space, then they are likely to form a link. Hence this idea should carry over to other social network models as well.

3 Deterministic Model with Identical Radii

Consider a simple version of the RHH model where all radii are equal to r , and $\alpha \rightarrow \infty$. This implies that two nodes i and j share a link (henceforth, $i \sim j$) iff the distance d_{ij} between them is constrained by $d_{ij} < r$. Thus, given node positions, links are deterministic; however the node positions are still non-deterministic. While this might appear to be a strong constraint, we will show later in Section 6 that similar results are applicable even for finite but large α . We now analyze the simplest of heuristics: counting the common neighbors of i and j . Let there be N nodes in total.

Let $\mathcal{N}(i)$ be the set of neighbors of node i . Let Y_k be a random variable which is 1 if $k \in \mathcal{N}(i) \cap \mathcal{N}(j)$, and 0 otherwise. Given d_{ij} , for all $k \notin \{i, j\}$, the Y_k 's are independent since they only depend on the position

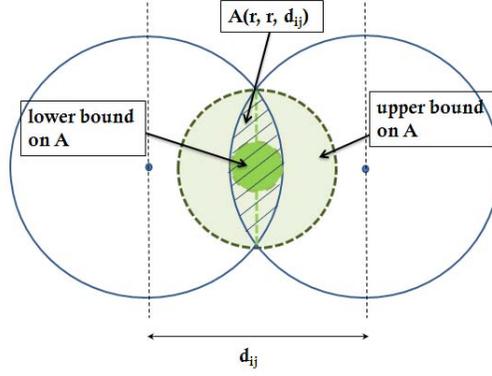


Figure 1: Common neighbors of two nodes must lie in the intersection $A(r, r, d_{ij})$.

of point k . Hence, we have

$$E[Y_k | d_{ij}] = P(i \sim k \sim j | d_{ij}) = \int_{d_{ik}, d_{jk}} P(i \sim k | d_{ik}) P(j \sim k | d_{jk}) P(d_{ik}, d_{jk} | d_{ij}) d(d_{ij}) \quad (1)$$

In the deterministic model, this quantity is exactly equal to the volume of intersection of two balls of radius r centered at i and j (see Figure 1). Denote this volume by $A(r, r, d_{ij})$. Also, the observed value of $\sum_k Y_k$ is simply the number of common neighbors η . From now on we will drop the d_{ij} part when we write expectation for notational convenience. However any expectation in terms of area of intersection is obviously computed given the pairwise distance d_{ij} . Thus by using empirical Bernstein bounds (Maurer & Pontil, 2009), we have:

$$P \left[\left| \sum_k Y_k / N - E[Y_k] \right| \geq \sqrt{\frac{2 \text{var}_N(Y) \log 2/\delta}{N}} + \frac{7 \log 2/\delta}{3(N-1)} \right] \leq 2\delta \quad (2)$$

$\text{var}_N(Y)$ is the sample variance of Y , i.e. $\frac{\eta(1 - \eta/N)}{N-1}$. Setting $\epsilon = \sqrt{\frac{2 \text{var}_N(Y) \log 2/\delta}{N}} + \frac{7 \log 2/\delta}{3(N-1)}$

$$P \left[\frac{\eta}{N} - \epsilon \leq A(r, r, d_{ij}) \leq \frac{\eta}{N} + \epsilon \right] \geq 1 - 2\delta \quad (3)$$

$A(r, r, d_{ij})$ is twice the volume of a spherical cap whose base is $d_{ij}/2$ distance away from the center:

$$A(r, r, d_{ij}) = 2 \frac{\pi^{\frac{D-1}{2}} r^D}{\Gamma(\frac{D+1}{2})} \int_0^{\cos^{-1}\left(\frac{d_{ij}}{2r}\right)} \sin^D(t) dt \quad (4)$$

Given the above bounds on $A(r, r, d_{ij})$, we can obtain bounds on d_{ij} by solving eq. (4) numerically. However, weaker analytic formulas can be obtained by using hyperspheres bounding this intersection, as shown in Figure 1. $V(r)$ is the volume of a D dimensional hypersphere of radius r .

$$\left(1 - \frac{d_{ij}}{2r}\right)^D \leq \frac{A(r, r, d_{ij})}{V(r)} \leq \left(1 - \left(\frac{d_{ij}}{2r}\right)^2\right)^{D/2} \quad (5)$$

Using this in eq. (3) gives us bounds on d_{ij} :

$$2r \left(1 - \left(\frac{\eta/N + \epsilon}{V(r)}\right)^{1/D}\right) \leq d_{ij} \leq 2r \sqrt{1 - \left(\frac{\eta/N - \epsilon}{V(r)}\right)^{2/D}}$$

Using common neighbors in link prediction. Let us recall that in link prediction, we want to pick the node which is most likely to be a neighbor of i , and is not currently a neighbor (call this OPT). If we knew the positions, we would pick the non-neighbor with the minimum distance (d_{OPT}). However, since positions in latent space are unknown, we instead predict a link to the node that shares the most common neighbors

with i (call this MAX). Here we will show that (Lemma 3.2) the distance to the node with largest common neighbors (d_{MAX}) is within an additive factor of d_{OPT} . This factor goes to zero as N increases. This again shows that, as N increases, link prediction using the number of common neighbors converges to the optimal prediction.

Let the number of common neighbors between i and OPT be η_{OPT} , and between i and MAX be η_{MAX} . Now we will try to relate d_{OPT} with d_{MAX} . Note that both these distances are larger than r . Denote the area of intersections of these two nodes with i as A_{OPT} and A_{MAX} respectively. Then, we have:

Lemma 3.1 Define $\epsilon_o = \sqrt{\frac{2\text{var}_N(Y_{OPT}) \log 2/\delta}{N}} + \frac{7 \log 2/\delta}{3(N-1)}$ and $\epsilon_m = \sqrt{\frac{2\text{var}_N(Y_{MAX}) \log 2/\delta}{N}} + \frac{7 \log 2/\delta}{3(N-1)}$, where Y_{OPT} and Y_{MAX} denote the random variable for common neighbors between i and OPT, and i and MAX respectively.

$$A_{OPT} \geq A_{MAX} \quad P[A_{MAX} \geq A_{OPT} - \epsilon_o - \epsilon_m] \geq 1 - 2\delta$$

Proof: Using the high probability bounds from eq. (3) we have (w.h.p),

$$A_{MAX} - A_{OPT} \geq \frac{\eta_{MAX}}{N} - \epsilon_m - \left(\frac{\eta_{OPT}}{N} + \epsilon_o\right)$$

By definition, $\eta_{MAX} \geq \eta_{OPT}$. This and the high probability empirical Bernstein bound on A_{OPT} yield the result. \blacksquare

This means that as N becomes large, the node with the highest number of common neighbors will be the optimal node for link prediction. Now we will give a bound on how far d_{OPT} is from d_{MAX} .

Theorem 3.2 Define $\epsilon_o = \sqrt{\frac{2\text{var}_N(Y_{OPT}) \log 2/\delta}{N}} + \frac{7 \log 2/\delta}{3(N-1)}$, $\epsilon_m = \sqrt{\frac{2\text{var}_N(Y_{MAX}) \log 2/\delta}{N}} + \frac{7 \log 2/\delta}{3(N-1)}$, and $\epsilon_f = \epsilon_o + \epsilon_m$.

$$d_{OPT} \leq d_{MAX} \stackrel{w.h.p}{\leq} d_{OPT} + 2r \left(\frac{\epsilon_f}{V(r)}\right)^{1/D} \leq d_{OPT} + 2 \left(\frac{\epsilon_f}{V(1)}\right)^{1/D}$$

4 Deterministic Model with Distinct Radii

Until now our model has used the same r for all nodes. The degree of a node is distributed as $Bin(N, V(r))$, where $V(r)$ is the volume of a radius r . Thus r determines the degree of a node in the graph, and identical r will lead to a roughly regular graph. In practice, social networks are far from regular. In order to accommodate complex networks we will now allow a different radius (r_i) for node i . For this section, we will assume that these radii are given to us. The new connectivity model is: $i \rightarrow j$ iff $d_{ij} \leq r_j$, where $i \rightarrow j$ now represents a *directed edge* from i to j . While variants of this are possible, this is similar in spirit to a citation network, where a paper i tends to cite a well-cited paper j (with larger number of in-neighbors) than another infrequently cited paper on the same topic; here, r_j can be thought of as the measure of popularity of node j . Under this model, we will show why some link prediction heuristics work better than others.

As in the previous section, we can use common neighbors to estimate distance between nodes. We can count common neighbors in 4 different ways as follows:

- (Type-1) All k , s.t. $k \rightarrow i$ and $k \rightarrow j$: all nodes which point to both i and j . The probability of this given d_{ij} is $P(d_{ik} \leq r_i \cap d_{jk} \leq r_j | d_{ij})$, which can be easily shown to be $A(r_i, r_j, d_{ij})$.
- (Type-2) All k , s.t. $i \rightarrow k$ and $j \rightarrow k$: all nodes to which both i and j point. The probability of this given d_{ij} is $A(r_k, r_k, d_{ij})$.
- (Type-3) All k , s.t. $i \rightarrow k$ and $k \rightarrow j$: all directed paths of length 2 from i to j . The probability of this given d_{ij} is given by $A(r_k, r_j, d_{ij})$.
- (Type-4) All k , s.t. $j \rightarrow k$ and $k \rightarrow i$: all directed paths of length 2 from j to i . The probability of this given d_{ij} is given by $A(r_i, r_k, d_{ij})$.

If we count type-1 nearest neighbors, the argument from section 3 carries over, and if there are enough common neighbors of this type, we can estimate d_{ij} by computing $A(r_i, r_j, d_{ij})$. However, if both r_i and r_j are small, there might not be many common neighbors; indeed, if $d_{ij} > r_i + r_j$, then there will be no type-1 common neighbors. In such cases, we consider type-2 neighbors, i.e. the ones which both i and j point to. The analysis for type-3 and type-4 neighbors is very similar to that for type-2, and hence we do not discuss

these any further. In the type-2 case, the radii r_k of the common neighbors play an important role. Intuitively, if both i and j point to a very popular node (high radius r_k), then that should not give us a lot of information about d_{ij} , since it is not very surprising. In particular, any type-2 common neighbor k leads to the following constraint: $d_{ij} \leq d_{ik} + d_{jk} \leq 2r_k$. Obviously, the bound is stronger for small values of r_k . This argues for weighting common neighbors differently, depending on their radii. We formalize this intuition using a toy example now. The analysis in the following section can be generalized to graphs, where the radii of the nodes form a finite set.

Recall that common neighbors can be expressed as a sum of random variables Y_k , which is 1, if k is a common neighbor of i and j .

Motivating Example. Take a toy network where the nodes can have two different radii R and R' , with $R < R'$. The total number of low radii nodes is N_R , whereas that of large radii nodes is $N_{R'}$.

The expectation of the number of type-2 common neighbors will now have a mixture of $A(R, R, d_{ij})$ and $A(R', R', d_{ij})$. One solution is to estimate high probability bounds on distances from the two different classes of common neighbors separately, and then examine the intersection of these bounds. We will discuss a new estimator based on this intuition at the end of this section. The other solution is to look at weighted combinations of common neighbors from different radii. The weights will reflect how important one common neighbor is relative to another. For example, consider a pair of papers which both cite a book on introduction to algorithms (cited by 5000 other papers, e.g. higher radius), and a specific article on randomized algorithms (cited by 30 other papers, e.g. lower radius). The second article gives more evidence on the ‘‘closeness’’ or similarity of the pair. We will consider this approach next.

Suppose we observe η_R common neighbors of N_R nodes of small radius, and $\eta_{R'}$ common neighbors of $N_{R'}$ nodes of large radius, between pair of nodes i, j . The likelihood of these observations, given the pairwise distance d_{ij} is:

$$P(\eta_R, N_R, \eta_{R'}, N_{R'} | d_{ij}) = \prod_{r \in \{R, R'\}} \binom{N_r}{\eta_r} A(r, r, d_{ij})^{\eta_r} (1 - A(r, r, d_{ij}))^{N_r - \eta_r} \quad (6)$$

We want to rank pairs of nodes using the distance estimate d^* , which maximizes the likelihood of this partial set of observations. However, if $\eta_R > 0$, the logarithm of the above is defined only when $d_{ij} \leq 2R$. To make the likelihood well-behaved, we introduce a small noise parameter β : node i connects to node j with probability $1 - \beta$ (if $d_{ij} \leq r_j$), or with probability β (otherwise). Now, the probability of having a type-2 common neighbor of radius r will be $\beta + A(r, r, d_{ij})(1 - \beta)$. For ease of exposition we will denote this by $A_\beta(r, r, d_{ij})$. The new likelihood will be exactly as in eq. (6), except we will use A_β instead of A . Setting the derivative of the logarithm yields:

$$w(R, d^*)N_R A_\beta(R, R, d^*) + w(R', d^*)N_{R'} A_\beta(R', R', d^*) = w(R, d^*)\eta_R + w(R', d^*)\eta_{R'} \quad (7)$$

where, $w(R, d^*) = \frac{-\frac{dA_\beta(R, R, d_{ij})}{d_{ij}} \Big|_{d^*}}{A_\beta(R, R, d^*)(1 - A_\beta(R, R, d^*))}$. Note that the negative sign is only to make both sides positive, since A_β decreases with distance.

Using Leibnitz’s rule on eq. 4, the derivative of A w.r.t d_{ij} can be written as:

$$A'(R, R, d_{ij}) = \frac{dA(R, R, d_{ij})}{d_{ij}} = \begin{cases} -C_D r^{D-1} \left(1 - \frac{d_{ij}^2}{4r^2}\right)^{\frac{D-1}{2}} & \text{If } d_{ij} \leq 2r \\ 0 & \text{Otherwise} \end{cases} \quad (8)$$

$$A'_\beta(R, R, d_{ij}) = \frac{dA_\beta(R, R, d_{ij})}{d_{ij}} = (1 - \beta)A'(R, R, d_{ij})$$

Suppose we could approximate $w(R, d_{ij})$ with some w_R that depends only on R and not on d_{ij} . Then, the RHS of eq. (7) can be obtained from the data alone. Also, it can be shown that the L.H.S. of eq. (8) is a monotonically decreasing function of d^* . Hence, a pair of nodes with higher RHS will have lower distance estimate, implying that ranking based on the RHS will be equivalent to ranking based on distance. All that remains is finding a good approximation w_R .

We start by bounding $A'(R, R, d_{ij})$ in terms of $A(R, R, d_{ij})$. Consider $D > 1^1$. Combining eq. (8) with eq. (5) yields a lower bound: $A'(R, R, d) \geq c'_D \frac{A}{\sqrt{r^2 - d^2/4}}$. For the upper bound, we note that the volume of the spherical cap can be lower bounded by the volume of a sphere in $D - 1$ dimensions of radius $\sqrt{r^2 - d^2/4}$,

¹When $D = 1$, A' is constant, and $A(r, r, d) = 2r - d$.

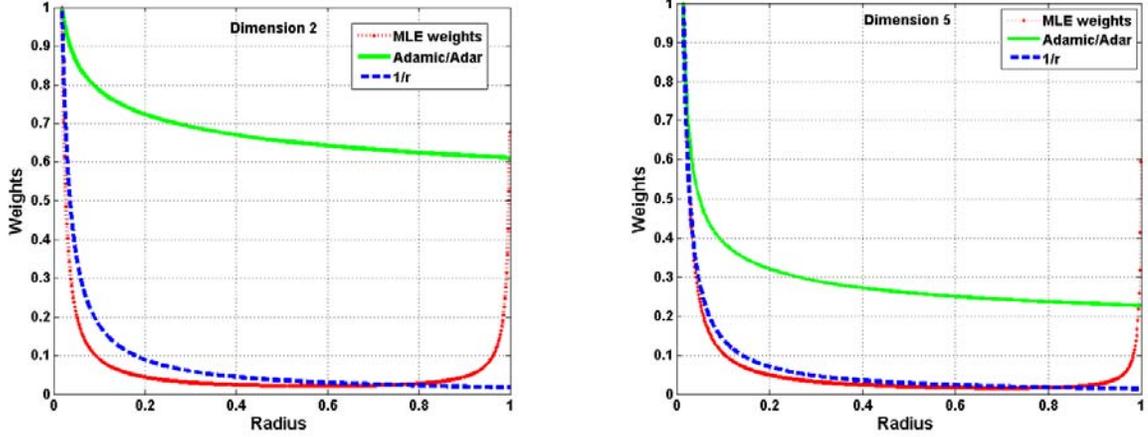


Figure 2: For a given distance d , we plot $w(r, d)$, *Adamic/Adar* and $1/r$ with increasing r . Note that both axes are scaled to a maximum of 1. Note that $1/r$ closely matches $w(r, d)$ for a wide range of radii.

times the height $d - 2r$, times a constant depending only on D : $A(r, r, d) \geq k_D \left(r^2 - \frac{d^2}{4}\right)^{\frac{D-1}{2}} \left(r - \frac{d}{2}\right)$
Combining with eq. (8) gives: $-A'(r, r, d) \leq k'_D \frac{A}{r-d/2}$. Given that β is extremely small, we have

$$\frac{1}{r} c''_D \frac{1}{\sqrt{1 - \frac{d^2}{4r^2}}} \lesssim \frac{|A'_\beta(r, r, d)|}{A_\beta(r, r, d)(1 - A_\beta(r, r, d))} \leq \frac{1}{r} k''_D \frac{1}{(1 - V(r))(1 - \frac{d}{2r})}$$

For a given distance d and increasing radius r , the weight $w(r, d)$ first decreases sharply but increases again once r becomes close to the maximum radius, i.e., $V(r) \approx 1$ (see Figure 2). Thus, it is high for both nodes of very low and very high radius. Clearly, the presence of a low-radius common neighbor gives strong evidence that d is small. On the other hand, the *absence* of a very high degree node gives strong evidence that d is very large. Note that, the presence of low radius common neighbors in the absence of very high radius common neighbors is extremely unlikely. This is because, if a pair of nodes are close enough to connect to a low radius node, they are also very likely to both be within the radius of some very high radius node.

Since $NV(r)$ is the expectation of the indegree of a node of radius r , such high-radius nodes are expected to have extremely high degrees. However, high-degree nodes in real-world settings typically connect to no more than 10 – 20% of the set of nodes, which is why a practical weighting only needs to focus on situations where $1 - V(r) \approx 1$. For relatively small d (which are the interesting candidates for link prediction), the weights $w(r, d)$ are then well approximated by $w(r) = 1/r$ up to a constant. Note that this is identical to weighting a node by $1/(NV(r))^{1/D}$, i.e., essentially weighting a common neighbor i by $1/\deg(i)^{1/D}$.

Now we will discuss a popular weighting scheme, namely *Adamic/Adar*, which weights the common neighbors by $1/\log(\deg(i))$.

Adamic/Adar: A popular link prediction heuristic. In practice, the radius of a node is analogous to its degree, and hence it is natural to weight a node more if it has lower degree. The *Adamic/Adar* measure (Adamic & Adar, 2003) was introduced to measure how related two home-pages are. The authors computed this by looking at common features of the webpages, and instead of computing just the number of such features, they weighted the rarer features more heavily. In our social networks context, this is equivalent to computing similarity between two nodes by computing the number of common neighbors, where each is weighted inversely by the logarithm of its degree.

$$\text{Adamic/Adar} = \sum_{k \in \mathcal{N}(i) \cap \mathcal{N}(j)} \frac{1}{\log(\deg(k))}$$

(Liben-Nowell & Kleinberg, 2003) have shown that this out-performs the number of common neighbors in a variety of social and citation networks, confirming the positive effect of a skewed weighting scheme that we observed in the motivating example. *Adamic/Adar* would be expected to perform relatively poorly only in case (iv), but since the weighting factor for a node k is only $1/\log(\deg(k))$, it takes exponentially high degrees for the skew to have a significant negative impact on performance.

We can analyze the Adamic/Adar measure as follows. In our model, the expected degree of a node k of radius r_k is simply $NV(r_k)$, so we set the weights as $w_k = 1/\log(NV(r_k))$. Let $\mathcal{S} = \sum_k w_k Y_k$, where random variable $Y_k = 1$ if k is a type-2 common neighbor of i and j , and zero otherwise. Clearly, $E[\mathcal{S}] = \sum_k w_k A(r_k, r_k, d_{ij}) = \sum_k A(r_k, r_k, d_{ij})/\log(NV(r_k))$. Let the minimum and maximum radii be r_{\min} and r_{\max} respectively. The following can be easily obtained from the Chernoff bound.

Lemma 4.1 $\frac{\mathcal{S}}{N} \left(1 - \sqrt{\frac{3 \log(NV(r_{\max})) \ln(1/\delta)}{N \cdot A(r_{\max}, r_{\max}, d_{ij})}}\right) \leq \frac{E[\mathcal{S}]}{N} \leq \frac{\mathcal{S}}{N} \left(1 + \sqrt{\frac{3 \log(NV(r_{\min})) \ln(1/\delta)}{N \cdot A(r_{\min}, r_{\min}, d_{ij})}}\right)$

Clearly, the error terms decay with increasing N , and for large N , we can tightly bound $E[\mathcal{S}]$. Since $E[\mathcal{S}]$ is monotonically decreasing function of d_{ij} , this translates into bounds on d_{ij} as well.

New estimators. Based on the analysis of the motivating example discussed before, we can get a lot of intuition about a general graph. We have seen that low radius common neighbors imply that distance is small, whereas fewer high degree common neighbors in the absence of any low degree common neighbors imply that distance is large. Based on these observations, we will define the following two estimators.

Consider the estimate Q_R , which is simply the fraction of nodes with radius smaller than R that are type-2 neighbors of i and j . Let N_R be the number of nodes with radius less than R . We define Q_R as follows:

$$Q_R = \frac{\sum_{r_k \leq R} Y_k}{N_R} \quad \rightarrow \quad E[Q_R] = \frac{\sum_{r_k \leq R} A(r_k, r_k, d_{ij})}{N_R} < A(R, R, d_{ij}) \quad (9)$$

Q_R is large when many low-radius nodes are type-2 common neighbors of i and j . Application of Hoeffding bounds give:

$$P \left[Q_R \leq E[Q_R] + \sqrt{\frac{1}{2N_R} \ln \left(\frac{1}{\delta} \right)} \right] \geq 1 - \delta$$

We pick R so that $E[Q_R]$ is *large enough*. While iterative algorithms can give an exact value of the upper bound on d_{ij} , we will also provide a simple intuitive bound:

$$\begin{aligned} Q_R &\leq A(R, R, d_{ij}) + \sqrt{\frac{1}{2N_R} \ln \left(\frac{1}{\delta} \right)} \leq V(R) \left(1 - \frac{d_{ij}^2}{4R^2}\right)^{D/2} + \sqrt{\frac{1}{2N_R} \ln \left(\frac{1}{\delta} \right)} \\ \Rightarrow d_{ij} &\leq 2R \sqrt{1 - \left(\frac{Q_R - \sqrt{\ln(1/\delta)/2N_R}}{V(R)} \right)^{2/D}} \end{aligned}$$

If we observe a large Q_R for a small R , then this upper bound gets smaller. This shows that a large number of neighbors of small degree gives a tighter upper-bound on d_{ij} . In the same spirit, we can define another estimator $T_{R'}$, such that

$$T_{R'} = \frac{\sum_{r_k \geq R'} Y_k}{N_{R'}} \quad \rightarrow \quad E[T_{R'}] = \frac{\sum_{r_k \geq R'} A(r_k, r_k, d_{ij})}{N_{R'}} \geq A(R', R', d_{ij})$$

A similar analysis yields a high probability lower-bound on d_{ij} :

$$\begin{aligned} T_{R'} &\geq A(R', R', d_{ij}) - \sqrt{\frac{1}{2N_{R'}} \ln \left(\frac{1}{\delta} \right)} \geq V(R') \left(1 - \frac{d_{ij}}{2R'}\right)^D - \sqrt{\frac{1}{2N_{R'}} \ln \left(\frac{1}{\delta} \right)} \\ \Rightarrow d_{ij} &\geq 2R' \left(1 - \left(\frac{T_{R'} + \sqrt{\ln(1/\delta)/2N_{R'}}}{V(R')} \right)^{1/D} \right) \end{aligned}$$

Smaller values of $T_{R'}$ yield tighter lower bounds. This tells us that if many high degree nodes are *not* common neighbors of i and j then, we are more confident that i and j are far away.

While Q_R and $T_{R'}$ could be used with any R and R' , we could also perform a sweep over the range of possible radii, computing bounds on d_{ij} for each radius using both estimators, and then retaining the best.

5 Estimators using Longer Paths in the Deterministic Model

The bounds on the distance d_{ij} described in the previous sections apply only when i and j have common neighbors. However, there will be no common neighbors if (for the undirected case) (a) $d_{ij} > 2r$, or (b) no points fall in the intersection area $A(r, r, d_{ij})$ due to small sample size N . In such cases, looking at paths of length $\ell > 2$ between i and j can yield bounds on d_{ij} . Such bounds can be useful even when common

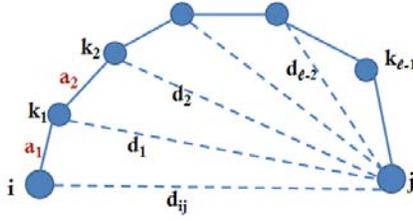


Figure 3: Triangulation for bounding d_{ij} using ℓ -hop paths.

neighbors exist; in fact, we show that the mere existence of *one* common neighbor leads to stronger bounds for long paths.

We first discuss how d_{ij} can be upper-bounded using the observed number of simple ℓ -hop paths for any given $\ell > 2$. A stronger upper bound and a lower bound can be derived when i and j are known to have at least one common neighbor. We demonstrate this for $\ell = 3$, with a similar technique being applicable for longer paths as well. For the sake of simplicity, we restrict ourselves to the case of identical and known r .

An Upper Bound for $\ell > 2$. Let $Y(i, k_1, \dots, k_{\ell-2}, j) = 1$ if there is a simple path of length ℓ such that $i \sim k_1 \sim k_2 \sim \dots \sim k_{\ell-2} \sim j$, with no node being repeated in the path. Let $\mathcal{S}^{(\ell)}$ be the set of ordered sets of ℓ distinct elements from $\{1, \dots, N\}$; thus, $\mathcal{S}^{(\ell)}$ represents all possible paths of length ℓ . Let $\eta_\ell(i, j | X_1, \dots, X_N)$ be the number of paths of length ℓ between i and j , given the positions of all N points:

$$\eta_\ell(i, j | X_1, \dots, X_N) = \sum_{k_1, \dots, k_{\ell-2} \in \mathcal{S}^{(\ell-2)}} Y(i, k_1, \dots, k_{\ell-2}, j).$$

We need to infer bounds on d_{ij} given the observed number of simple paths $\eta_\ell(i, j)$. Our first step is to bound the maximum degree Δ of any graph generated by the RHH model. Next, we use Δ to bound both the maximum possible value of $\eta_\ell(i, j)$ and the change that can be induced in it by moving any one point. Finally, we compute the expected value $E[\eta_\ell(i, j)]$. Combining these will give us a bound linking d_{ij} to the number of simple ℓ -hop paths.

Under the assumption that the positions X_k are uniformly distribution in the unit hypersphere, we can bound Δ , as follows:

Lemma 5.1 $\Delta < NV \left(1 + \sqrt{\frac{\ln(N/\delta)}{2NV}} \right)$ with probability at least $1 - \delta$.

Proof: The degree $d(k)$ of any node k is a binomial random variable with expectation $E[d(k)] = NV$, where V is the volume of a hypersphere of radius r . Thus, using the Chernoff bound, $d(k) < NV \left(1 + \sqrt{\frac{\ln(N/\delta)}{2NV}} \right)$ holds with probability at least $(1 - \delta/N)$. Applying the union bound on all nodes yields the desired proposition. \blacksquare

Lemma 5.2 For any graph with maximum degree Δ , we have: $\eta_\ell(i, j) \leq \Delta^{\ell-1}$.

Proof: This can be proved using a simple inductive argument. If the graph is represented by adjacency matrix \mathbf{M} , then the number of length ℓ paths between i and j is given by $\mathbf{M}^\ell(i, j)$. Trivially \mathbf{M}_{ij}^2 can be at most Δ . This happens when both i and j have degree Δ , and their neighbors form a perfect matching. Assuming this is true for all $m < \ell$, we have: $\mathbf{M}^\ell(i, j) = \sum_p \mathbf{M}(i, p) \mathbf{M}^{\ell-1}(p, j) \leq \Delta^{\ell-2} \sum_p \mathbf{M}(i, p) \leq \Delta^{\ell-1}$. \blacksquare

Lemma 5.3 For $\ell < \Delta$, $|\eta_\ell(i, j | X_1, \dots, X_p, \dots, X_N) - \eta_\ell(i, j | X_1, \dots, \tilde{X}_p, \dots, X_N)| \leq (\ell - 1) \cdot \Delta^{\ell-2}$

Proof: The largest change in $\eta_\ell(\cdot)$ occurs when node p was originally unconnected to any other node, and is moved to a position where it can maximally add to the number of ℓ -hop paths between i and j (or vice versa). Consider all paths where p is m hops from i (and hence $\ell - m$ hops from j). From Lemma 5.2, the number of such paths can be at most $\Delta^{m-1} \cdot \Delta^{\ell-m-1} = \Delta^{\ell-2}$. Since $m \in \{1, \dots, \ell - 1\}$, the maximum change is $(\ell - 1) \cdot \Delta^{\ell-2}$. \blacksquare

The bounds in both Lemma 5.2 and 5.3 are tight, as can be seen by considering a clique of i, j , and $\Delta - 1$ other nodes. Next, we compute the expected number of ℓ -hop paths. Define $A(r_1, r_2, d)$ as the volume of intersection of two balls of radii r_1 and r_2 , whose centers are distance d apart (as always, the dimension D is implicit). Define $Pr_\ell(i, j)$ as the probability of observing an ℓ -hop path between points i and j .

Theorem 5.4 $E[\eta_\ell(i, j)] \leq \Delta^{\ell-1} \prod_{p=1}^{\ell-1} A(r, p \times r, (d_{ij} - (\ell - p - 1)r)_+)$, where x_+ is defined as $\max(x, 0)$.

Proof: Consider an ℓ -hop path between nodes i and j as in figure 3. The solid lines are the edges in the path, whereas the dotted lines are the distance variables we will introduce in order to compute the probability of this path. For clarity of notation, let us denote the distances d_{ik_1}, d_{ik_2} etc. by a_1, a_2 , up to $a_{\ell-1}$. These are all less than r , since the corresponding edges are present. We also denote the distances d_{jk_1}, d_{jk_2} , etc. by d_1, d_2 , up to $d_{\ell-1}$. From the triangle inequality, $d_{\ell-2} \leq a_{\ell-1} + a_\ell \leq 2r$, and by induction, $d_k \leq (\ell - k)r$. Similarly, $d_1 \geq (d_{ij} - a_1)_+ \geq (d_{ij} - r)_+$, and by induction, $d_k \geq (d_{ij} - kr)_+$. We can now compute the probability of observing an ℓ -hop path:

$$\begin{aligned}
Pr_\ell(i, j) &= P(i \sim k_1 \sim \dots \sim k_{\ell-1} \sim j | d_{ij}) \\
&= P(a_1 \leq r \cap \dots \cap a_\ell \leq r | d_{ij}) \\
&= \int_{d_1, \dots, d_{\ell-2}} P(a_1 \leq r, \dots, a_{\ell-1} \leq r, d_1, \dots, d_{\ell-2} | d_{ij}) \\
&= \int_{d_1=(d_{ij}-r)_+}^{(\ell-1)r} \dots \int_{d_{\ell-2}=(d_{ij}-(\ell-2)r)_+}^{2r} P(a_{\ell-1} \leq r, a_\ell \leq r | d_{k_{\ell-2}}) P(a_{\ell-2} \leq r, d_{\ell-2} | d_{k_{\ell-3}}) \dots P(a_1 \leq r, d_1 | d_{ij}) \\
&\leq A(r, r, (d_{ij} - (\ell - 2)r)_+) \times A(r, 2r, (d_{ij} - (\ell - 3)r)_+) \times \dots \times A(r, (\ell - 1)r, d_{ij}) \\
&\leq \prod_{p=1}^{\ell-1} A(r, p \times r, (d_{ij} - (\ell - p - 1)r)_+) \tag{10}
\end{aligned}$$

Since there can be at most $\Delta^{\ell-1}$ possible paths (from Lemma 5.2), the theorem statement follows. \blacksquare

Corollary 5.5 For 3-hop paths, we have:

$$Pr_3(i, j) \leq A(r, r, (d_{ij} - r)_+) \cdot A(r, 2r, d_{ij}) \quad \text{AND} \quad E[\eta_3(i, j)] \leq \Delta^2 \cdot A(r, r, (d_{ij} - r)_+) \cdot A(r, 2r, d_{ij})$$

Theorem 5.6

$$\eta_\ell(i, j) \leq (NV)^{\ell-1} \left[\prod_{p=1}^{\ell-1} A(r, p \times r, (d_{ij} - (\ell - p - 1)r)_+) + \frac{(\ell - 1) \sqrt{\frac{1/\delta}{2}}}{\sqrt{NV} \left(1 + \sqrt{\frac{\ln(N/\delta)}{2NV}}\right)} \right] \left(1 + \sqrt{\frac{\ln(N/\delta)}{2NV}}\right)^{\ell-1}$$

with probability at least $(1 - 2\delta)$.

Proof: From McDiarmid's inequality (McDiarmid, 1989), we have:

$$\begin{aligned}
\eta_\ell(i, j) &\leq E[\eta_\ell(i, j)] + (\ell - 1) \Delta^{\ell-2} \sqrt{\frac{N \ln(1/\delta)}{2}} \\
&\leq \Delta^{\ell-2} \left[\Delta \prod_{p=1}^{\ell-1} A(r, p \times r, (d_{ij} - (\ell - p - 1)r)_+) + (\ell - 1) \sqrt{\frac{N \ln(1/\delta)}{2}} \right] \\
&\leq (NV)^{\ell-1} \left[\prod_{p=1}^{\ell-1} A(r, p \times r, (d_{ij} - (\ell - p - 1)r)_+) + \frac{(\ell - 1) \sqrt{\frac{\ln(1/\delta)}{2}}}{\sqrt{NV} \left(1 + \sqrt{\frac{\ln(N/\delta)}{2NV}}\right)} \right] \left(1 + \sqrt{\frac{\ln(N/\delta)}{2NV}}\right)^{\ell-1}
\end{aligned}$$

where Theorem 5.4 is applied in the step 2, and Lemma 5.1 in step 3. Note that as N increases, the second term in the summation decays, yielding tighter bounds. \blacksquare

Bounding d_{ij} . Theorem 5.6 yields an upper bound d_{ij} as follows. Only the first term in the summation depends on d_{ij} , and this term decreases monotonically with increasing d_{ij} . Thus, a simple binary search can give us the value of d_{ij} that achieves the equality in Theorem 5.6, and this is an upper bound on d_{ij} .

A looser but analytic bound can be obtained by upper-bounding all but one of the $A(\cdot)$ terms by 1. For example, using $A(r, 2r, d_{ij}) \leq 1$ in Corollary 5.5 yields $E[\eta_3(i, j)] \leq \Delta^2 A(r, r, (d_{ij} - r)_+)$. Using this in McDiarmid's inequality yields a bound of the form

$$A(r, r, (d_{ij} - r)_+) \geq \frac{\eta_3(i, j)}{c(N, \delta)} - c'(N, \delta) \Rightarrow d_{ij} \leq r + 2r \sqrt{1 - \left(\frac{\eta_3(i, j)/c(N, \delta) - c'(N, \delta)}{V(r)} \right)^{2/D}}$$

In general, bounds for ℓ -hop paths are of the form $d_{ij} \leq \ell r(1 - g(\eta_\ell(i, j), \epsilon))$. Thus, for some $\ell' > \ell$, $\eta_{\ell'}(i, j)$ needs to be much larger than $\eta_\ell(i, j)$ for the bound using ℓ' to be stronger than that for ℓ . In particular, this shows that when enough common neighbors are present (i.e., 2-hop paths), looking at longer paths is unlikely to improve bounds and help link prediction, thus theoretically confirming the empirical observations of (Liben-Nowell & Kleinberg, 2003).

Better bounds when shorter paths exist. While the above applies in the general case, it suffers from two weaknesses: (1) the upper bound on $E[\eta_\ell(i, j)]$ (and hence on d_{ij}) derived in Theorem 5.4 gets progressively weaker as ℓ increases, and (2) we could not derive a useful lower bound on $E[\eta_\ell(i, j)]$ (the trivial lower bound is zero, which is achieved when $d_{ij} = \ell r$). Both of these can be remedied if we know that *at least* one path of length less than ℓ exists. We demonstrate the idea for $\ell = 3$, but it can be used for larger ℓ in a similar fashion. First, we prove two bounds on the probability $Pr_3(i, j)$ of observing a 3-hop path between two points whose distance is d_{ij} .

Lemma 5.7 *If there exists any 3-hop path between i and j , then, for any $d' \in [(d_{ij} - 2r)_+, 2r]$,*

$$A(r, d', d_{ij}) \cdot A(r, r, d') \leq \mathbf{Pr}_3(\mathbf{i}, \mathbf{j}) \leq [A(r, r, (d_{ij} - r)_+) - A(r, r, d')] A(r, d', d_{ij}) + A(r, r, d') \cdot A(r, 2r, d_{ij})$$

Proof: Consider all points that are within distance r of point i , and within a distance range $(x, x + \Delta x)$ of point j ; here, Δx refers to an infinitesimal change in x . Since these points are within r of i , they can be the first hop in a 3-hop path from i to j . Also, since they are all equidistant from j , and the probability of a common neighbor between two points depends only on their distance, all of these points have the same probability $A(r, x, d_{ij})$ of forming a common neighbor with j . Let $p(r, x, d_{ij})$ denote the probability density function for such points. Triangle inequalities imply that $2r \geq x \geq (d_{ij} - r)_+$ for any 3-hop path to exist. Then,

$$\begin{aligned} Pr_3(i, j) &= \int_{(d_{ij}-r)_+}^{2r} p(r, x, d_{ij}) \cdot A(r, r, x) dx \geq \int_{(d_{ij}-r)_+}^{d'} p(r, x, d_{ij}) \cdot A(r, r, x) dx \\ &\geq A(r, r, d') \int_{(d_{ij}-r)_+}^{d'} p(r, x, d_{ij}) dx \geq A(r, d', d_{ij}) \cdot A(r, r, d') \end{aligned}$$

This proves the lower-bound on $Pr_3(i, j)$. The proof for the upper bound is similar:

$$\begin{aligned} Pr_3(i, j) &= \int_{(d_{ij}-r)_+}^{d'} p(r, x, d_{ij}) \cdot A(r, r, x) dx + \int_{d'}^{2r} p(r, x, d_{ij}) \cdot A(r, r, x) dx \\ &\leq A(r, r, (d_{ij} - r)_+) \cdot A(r, d', d_{ij}) + A(r, r, d') \cdot [A(r, 2r, d_{ij}) - A(r, d', d_{ij})] \\ &\leq [A(r, r, (d_{ij} - r)_+) - A(r, r, d')] A(r, d', d_{ij}) + A(r, r, d') \cdot A(r, 2r, d_{ij}) \end{aligned}$$

The difficulty in using these bounds arises from the fact that $d' \in [(d_{ij} - r)_+, 2r]$, and d_{ij} is unknown. When we only know that *some* 3-hop path exists, then $d_{ij} \leq 3r$, and hence $d' = 2r$ is the only value of d' that is guaranteed to lie within the required interval. In fact, using $d' = 2r$ yields exactly the statement of Corollary 5.5. However, suppose we also knew that at least one 2-hop path exists. Then, $d_{ij} \leq 2r$, and so any d' in the range $r \leq d' \leq 2r$ is valid. In particular, we can use $d' = r$ to get the following bounds. ■

Theorem 5.8 *When $d_{ij} \leq 2r$, then*

$$A(r, r, d_{ij}) \cdot A(r, r, r) \leq \mathbf{Pr}_3(\mathbf{i}, \mathbf{j}) \leq [A(r, r, (d_{ij} - r)_+) - A(r, r, r)] A(r, r, d_{ij}) + A(r, r, r) \cdot A(r, 2r, d_{ij})$$

Lemma 5.9 *The upper bound in Theorem 5.8 is smaller (and hence, better) than the upper bound in Corollary 5.5.*

Proof: The proof follows from basic algebraic manipulations. ■

Thus, the presence of even one common neighbor between i and j offers better bounds using 3-hop paths. In addition, we also have a lower bound that was unavailable in the general case. These translate to sharper bounds on d_{ij} , which can be obtained via binary search as described previously. Similar results can be obtained for paths of length $\ell > 3$.

Observations. Our analysis of ℓ -hop paths yields the following observations. (1) When short paths are non-existent or rare, the bounds on d_{ij} that we obtain through them can be loose. Longer paths can be used to yield better bounds in such cases. (2) As ℓ increases, more and more long paths need to be observed before the corresponding bound on d_{ij} becomes comparable or better than bounds obtained via shorter paths. (3) Even the *existence* of a short path can improve upper bounds obtained by all longer paths. In addition, lower bounds on d_{ij} can also be obtained. (4) The *number* of paths is important to the bound. Link prediction using the length of the shortest path ignores this information, and hence should perform relatively poorly, as observed by (Liben-Nowell & Kleinberg, 2003; Brand, 2005) and (Sarkar & Moore, 2007).

6 The Non-deterministic Case

All of the previous sections have assumed that, given the positions of points, the corresponding graph could be inferred exactly. In terms of the RHH model introduced in section 2, this corresponds to setting $\alpha \rightarrow \infty$. In this section, we investigate the effects of finite α . Our analysis shows that while bounds become looser, the results are still qualitatively similar.

The core idea underlying almost all of our previous results has been the computation of the probability of two nodes i and j having a common neighbor. For the deterministic case, this is simply the area of intersection of two hyperspheres, $A(r, r, d_{ij})$, for the case when all nodes have the same radius r . However, in the non-deterministic case, this probability is hard to compute exactly. Instead, we can give the following simple bounds on $Pr_2(i, j)$, which is the probability of observing a common neighbor between two nodes i and j that are distance d_{ij} apart and have identical radius r .

Theorem 6.1

$$Pr_2(i, j) > \frac{1}{4} (A(r, r, d_{ij}) + 2e^{-\alpha d_{ij}} \cdot (V(r) - A(r, r, d_{ij})))$$

$$Pr_2(i, j) < \begin{cases} A(r, r, d_{ij}) + 2V(r) \cdot \frac{\left[1 - \left(\frac{D}{\alpha r}\right)^D\right]}{\frac{\alpha r}{D} - 1} & (\text{for } \alpha r > D) \\ A(r, r, d_{ij}) + 2D \cdot V(r) & (\text{for } \alpha r = D) \\ A(r, r, d_{ij}) + 2V(D/\alpha) \cdot \frac{\left[1 - \left(\frac{\alpha r}{D}\right)^D\right]}{1 - \frac{\alpha r}{D}} & (\text{for } \alpha r < D) \end{cases}$$

Observations and Extensions. The importance of theorem 6.1 is that the probability of observing a common neighbor is still mostly dependent on the area of intersection of two hyperspheres, i.e. $A(r, r, d_{ij})$. However, there is a gap of a factor of 4 between the lower and upper bounds. This can still be used to obtain reasonable bounds on d_{ij} when enough common neighbors are observed. However, when we consider longer paths, the gap increases and we might no longer be able to get strong bounds.

The reason for this is that theorem 6.1 only uses the fact that probability of linking i and j is at least $1/2$ when d_{ij} is less than r . This statement is applicable to all α . However, we typically want to perform link prediction only when α is large, as small values of α yield graphs that are close to random and where no link prediction methods would work. For the case of large α , we can get much stronger lower bounds and close the factor-of-4 gap, as follows.

In order to compute the probability $Pr_2(i, j)$, we need to integrate the product of the link probabilities over the intersection of the two hyperspheres of radius r around nodes i and j . Let this region be denoted by $S(i, j)$. Suppose that, instead of integrating over $S(i, j)$, we integrate over a smaller subset $S'(i, j)$. While the volume of $S'(i, j)$ would be smaller, the minimum probabilities inside that subset could be much higher, leading to a better overall lower-bound. We consider $S'(i, j) = \{x_k | d_{ik} < r', d_{jk} < r'\}$ to be the intersection

of two hyperspheres of radius $r' < r$, centered on i and j . Then, using eq. 4

$$\begin{aligned} Pr_2(i, j, x_k \in S'(i, j)) & \\ & \geq \left(\frac{1}{1+e^{\alpha(r'-r)}} \right)^2 \cdot \text{vol}(S'(i, j)) \\ & \geq \left(\frac{1}{1+e^{\alpha(r'-r)}} \right)^2 V(r') \int_0^{\cos^{-1}\left(\frac{d_{ij}}{2r'}\right)} \sin^D(t) dt \end{aligned}$$

Ideally, we would like to pick r' to maximize this, but $\text{vol}(S'(i, j))$ depends on d_{ij} as well. Noting that and the effect of d_{ij} is restricted to the last term only, we propose the following formulation:

$$\text{Pick } r' \text{ to maximize } \left(\frac{1}{1+e^{\alpha(r'-r)}} \right)^2 \cdot V(r') \quad (11)$$

Lemma 6.2 *If $\alpha > D/r$, then $r' < r$.*

Thus, for large enough α , we can find a good r' which can improve the gap between upper and lower bounds of $Pr_2(i, j)$. The optimal r' gets closer to r as α increases, but its exact value has to be obtained numerically.

7 Summary and Discussion

The paper presents, to our knowledge, the first theoretical study of link prediction and the heuristics commonly used for that purpose. We formalize the link prediction problem as one of estimating distances between nodes in a latent space, where the observed graph structure provides evidence regarding the unobserved positions of nodes in this space. We present theoretical justifications of two common empirical observations: (1) the simple heuristic of counting common neighbors often outperforms more complicated heuristics, (2) a variant that weights common neighbors by the inverse of the logarithm of their degrees (Adamic & Adar, 2003) often performs better. We show that considering longer paths is useful only if shorter paths (especially, common neighbors) are not numerous enough for the bounds obtained from them to be tight enough. However, the bounds obtained from longer paths can be made significantly tighter if even a single short path is known to exist.

References

- Adamic, L. A., & Adar, E. (2003). Friends and neighbors on the web. *Social Networks*, 25.
- Aldous, D., & Fill, J. A. (2001). *Reversible markov chains*.
- Brand, M. (2005). A Random Walks Perspective on Maximizing Satisfaction and Profit. *SIAM '05*.
- Faust, K. (1988). Comparison of methods for positional analysis: Structural and general equivalences. *Social Networks*.
- Jeh, G., & Widom, J. (2002). Scaling personalized web search. *Stanford University Technical Report*.
- Katz, L. (1953). A new status index derived from sociometric analysis. *Psychometrika*.
- Kleinberg, J. (2000). The Small-World Phenomenon: An Algorithmic Perspective. *STOC*.
- Liben-Nowell, D., & Kleinberg, J. (2003). The link prediction problem for social networks. *CIKM '03*.
- Maurer, A., & Pontil, M. (2009). Empirical bernstein bounds and sample-variance penalization. *Conference on Learning Theory*.
- McDiarmid, C. (1989). On Method of Bounded Differences. *Surveys in Combinatorics*.
- McFarland, D. D., & Brown, D. J. (1973). Social distance as a metric: a systematic introduction to smallest space analysis. *Bonds of Pluralism: The Form and Substance of Urban Social Networks*.
- McPherson, M., Smith-Lovin, L., & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*.
- Raftery, A. E., Handcock, M. S., & Hoff, P. D. (2002). Latent space approaches to social network analysis. *J. Amer. Stat. Assoc.*, 15, 460.
- Raghavan, P. (2002). Social networks: From the web to the enterprise. *IEEE Internet Computing*.
- Sarkar, P., & Moore, A. (2007). A tractable approach to finding closest truncated-commute-time neighbors in large graphs. *Proc. UAI*.
- Schroeder, J., Xu, J. J., & Chen, H. (2003). Crimelink explorer: Using domain knowledge to facilitate automated crime association analysis. *ISI* (pp. 168–180).

The Convergence Rate of AdaBoost

Robert E. Schapire

Princeton University

Department of Computer Science

schapire@cs.princeton.edu

Abstract. We pose the problem of determining the rate of convergence at which AdaBoost minimizes exponential loss.

Boosting is the problem of combining many “weak,” high-error hypotheses to generate a single “strong” hypothesis with very low error. The AdaBoost algorithm of Freund and Schapire (1997) is shown in Figure 1. Here we are given m labeled training examples $(x_1, y_1), \dots, (x_m, y_m)$ where the x_i 's are in some domain \mathcal{X} , and the labels $y_i \in \{-1, +1\}$. On each round t , a distribution D_t is computed as in the figure over the m training examples, and a weak hypothesis $h_t : \mathcal{X} \rightarrow \{-1, +1\}$ is found, where our aim is to find a weak hypothesis with low weighted error ϵ_t relative to D_t . In particular, for simplicity, we assume that h_t minimizes the weighted error over all hypotheses belonging to some finite class of weak hypotheses $\mathcal{H} = \{\tilde{h}_1, \dots, \tilde{h}_N\}$.

The final hypothesis H computes the sign of a weighted combination of weak hypotheses $F(x) = \sum_{t=1}^T \alpha_t h_t(x)$. Since each h_t is equal to \tilde{h}_{j_t} for some j_t , this can also be rewritten as $F(x) = \sum_{j=1}^N \lambda_j \tilde{h}_j(x)$ for some set of values $\boldsymbol{\lambda} = \langle \lambda_1, \dots, \lambda_N \rangle$. It was observed by Breiman (1999) and others (Freund & Downs, 1998; Friedman et al., 2000; Mason et al., 1999; Onoda et al., 1998; Rätsch et al., 2001; Schapire & Singer, 1999) that AdaBoost behaves so as to minimize the exponential loss

$$L(\boldsymbol{\lambda}) = \frac{1}{m} \sum_{i=1}^m \exp \left(- \sum_{j=1}^N \lambda_j y_i \tilde{h}_j(x_i) \right)$$

over the parameters $\boldsymbol{\lambda}$. In particular, AdaBoost performs coordinate descent, on each round choosing a single coordinate j_t (corresponding to some weak hypothesis $h_t = \tilde{h}_{j_t}$) and adjusting it by adding α_t to it: $\lambda_{j_t} \leftarrow \lambda_{j_t} + \alpha_t$. Further, AdaBoost is greedy, choosing j_t and α_t so as to cause the greatest decrease in the exponential loss.

In general, the exponential loss need not attain its minimum at any finite $\boldsymbol{\lambda}$ (that is, at any $\boldsymbol{\lambda} \in \mathbb{R}^N$). For instance, for an appropriate choice of data (with $N = 2$ and $m = 3$), we might have

$$L(\lambda_1, \lambda_2) = \frac{1}{3} (e^{\lambda_1 - \lambda_2} + e^{\lambda_2 - \lambda_1} + e^{-\lambda_1 - \lambda_2}).$$

The first two terms together are minimized when $\lambda_1 = \lambda_2$, and the third term is minimized when $\lambda_1 + \lambda_2 \rightarrow +\infty$. Thus, the minimum of L in this case is attained when we fix $\lambda_1 = \lambda_2$, and the two weights together grow to infinity at the same pace.

Let $\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2, \dots$ be the sequence of parameter vectors computed by AdaBoost in the fashion described above. It is known that AdaBoost asymptotically converges to the minimum possible exponential loss (Collins et al., 2002). That is,

$$\lim_{t \rightarrow \infty} L(\boldsymbol{\lambda}^t) = \inf_{\boldsymbol{\lambda} \in \mathbb{R}^N} L(\boldsymbol{\lambda}).$$

However, it seems that only extremely weak bounds are known on the rate of convergence, for the most general case. In particular, Bickel, Ritov and Zakai (2006) prove a very weak bound of the form $O(1/\sqrt{\log t})$ on this rate. Much better bounds are proved by Rätsch, Mika and Warmuth (2002) using results from Luo and Tseng (1992), but these appear to require that the exponential loss be minimized by a finite $\boldsymbol{\lambda}$, and also depend on quantities that are not easily measured. Shalev-Shwartz and Singer (2008) prove bounds for a variant of AdaBoost. Zhang and Yu (2005) also give rates of convergence, but their technique requires a bound on the step sizes α_t . Many classic results are known on the convergence of iterative algorithms generally (see for instance, Luenberger and Ye (2008), or Boyd and Vandenberghe (2004)); however, these typically start by assuming that the minimum is attained at some finite point in the (usually compact) space of interest.

When the weak learning assumption holds, that is, when it is assumed that the weighted errors ϵ_t are all upper bounded by $1/2 - \gamma$ for some $\gamma > 0$, then it is known (Freund & Schapire, 1997; Schapire & Singer, 1999) that the exponential loss is at most $e^{-2t\gamma^2}$ after t rounds, so it clearly quickly converges to the

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
space $\mathcal{H} = \{h_1, \dots, h_N\}$ of weak hypotheses $h_j : \mathcal{X} \rightarrow \{-1, +1\}$
Initialize: $D_1(i) = 1/m$ for $i = 1, \dots, m$.
For $t = 1, \dots, T$:

- Train weak learner using distribution D_t ; that is, find weak hypothesis $h_t \in \mathcal{H}$ that minimizes the weighted error $\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i]$.
- Choose $\alpha_t = \frac{1}{2} \ln((1 - \epsilon_t)/\epsilon_t)$.
- Update, for $i = 1, \dots, m$: $D_{t+1}(i) = D_t(i) \exp(-\alpha_t y_i h_t(x_i)) / Z_t$
where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right)$.

Figure 1: The boosting algorithm AdaBoost.

minimum possible loss in this case. However, here our interest is in the general case when the weak learning assumption might not hold.

This problem of determining the rate of convergence is relevant in the proof of the consistency of AdaBoost given by Bartlett and Traskin (2007), where it has a direct impact on the rate at which AdaBoost converges to the Bayes optimal classifier (under suitable assumptions).

We conjecture that there exists a positive constant c and a polynomial $\text{poly}()$ such that for all training sets and all finite sets of weak hypotheses, and for all $B > 0$,

$$L(\lambda^t) \leq \min_{\lambda: \|\lambda\|_1 \leq B} L(\lambda) + \frac{\text{poly}(\log N, m, B)}{t^c}.$$

Said differently, the conjecture states that the exponential loss of AdaBoost will be at most ε more than that of any other parameter vector λ of ℓ_1 -norm bounded by B in a number of rounds that is bounded by a polynomial in $\log N, m, B$ and $1/\varepsilon$. (We require $\log N$ rather than N since the number of weak hypotheses $N = |\mathcal{H}|$ will typically be extremely large.) The open problem is to determine if this conjecture is true or false, in general, for AdaBoost. The result should be general and apply in all cases, even when the weak learning assumption does not hold, and even if the minimum of the exponential loss is not realized at any finite vector λ . The prize for a new result proving or disproving the conjecture is US\$100.

References

- Bartlett, P. L., & Traskin, M. (2007). AdaBoost is consistent. *Journal of Machine Learning Research*, 8, 2347–2368.
- Bickel, P. J., Ritov, Y., & Zakai, A. (2006). Some theory for generalized boosting algorithms. *Journal of Machine Learning Research*, 7, 705–732.
- Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.
- Breiman, L. (1999). Prediction games and arcing classifiers. *Neural Computation*, 11, 1493–1517.
- Collins, M., Schapire, R. E., & Singer, Y. (2002). Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48.
- Frean, M., & Downs, T. (1998). *A simple cost function for boosting* (Technical Report). Department of Computer Science and Electrical Engineering, University of Queensland.
- Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55, 119–139.
- Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 38, 337–374.
- Luenberger, D. G., & Ye, Y. (2008). *Linear and nonlinear programming*. Springer. Third edition.
- Luo, Z. Q., & Tseng, P. (1992). On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72, 7–35.
- Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). Functional gradient techniques for combining hypotheses. In *Advances in large margin classifiers*. MIT Press.
- Onoda, T., Rätsch, G., & Müller, K.-R. (1998). An asymptotic analysis of AdaBoost in the binary classification case. *Proceedings of the 8th International Conference on Artificial Neural Networks* (pp. 195–200).
- Rätsch, G., Mika, S., & Warmuth, M. K. (2002). On the convergence of leveraging. *Advances in Neural Information Processing Systems 14*.
- Rätsch, G., Onoda, T., & Müller, K.-R. (2001). Soft margins for AdaBoost. *Machine Learning*, 42, 287–320.
- Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37, 297–336.
- Shalev-Shwartz, S., & Singer, Y. (2008). On the equivalence of weak learnability and linear separability: New relaxations and efficient boosting algorithms. *21st Annual Conference on Learning Theory*.
- Zhang, T., & Yu, B. (2005). Boosting with early stopping: Convergence and consistency. *Annals of Statistics*, 33, 1538–1579.

Learning Talagrand DNF Formulas

Homin K. Lee

University of Texas at Austin
homin@cs.utexas.edu

Consider the following version of Talagrand’s probabilistic construction of a monotone function $f : \{0, 1\}^n \rightarrow \{0, 1\}$. Let f be an n -term monotone DNF formula where each term is selected independently and uniformly at random (with replacement) from the set of all $n^{\Omega(\log \log n)}$ possible terms of length $\log(n)$ over the first $\log^2(n)$ variables. Let us call such a DNF formula a *Talagrand DNF formula*. This is a scaled-down version of the construction by Talagrand (1996) which is defined over all n variables and has subexponentially many terms. I am interested in the following problem:

Does there exist a polynomial-time algorithm for learning the class of Talagrand DNF formulas over the uniform distribution on $\{0, 1\}^n$ given uniform random examples?

Ideally, I am looking for algorithms that will learn the class of *all* possible Talagrand DNF formulas in the worst-case. However, an average-case learning algorithm that succeeds with high probability over the choice of the Talagrand DNF formula as described above would be of significant interest as well.

Motivation: This problem is of course a special case of the corresponding learning problem for general polynomial-size DNF formulas. The problem of learning polynomial-size DNF formulas *without* queries has been open for almost twenty years (Valiant, 1984) and there has been no significant progress on the question until the last couple years.

Current Status: Recently, *random* DNF formulas were shown to be learnable in the following sequence of work (Jackson & Servedio, 2006; Jackson et al., To appear; Sellie, 2008; Sellie, 2009). The algorithms for learning random DNF formulas only work when the terms are well-separated, *i.e.*, when the terms share very few variables, which is clearly not the case for Talagrand DNF formulas.

Some Observations:

1. Unlike the class of all polynomial-size DNF formulas, the class of Talagrand DNF formulas as defined above do not suffer from the “junta” problem (Blum, 2003). We know exactly which variables are relevant.
2. The Talagrand functions are sensitive to noise as small as $1/\log(n)$, *i.e.*, $\Pr[f(x) \neq f(y)] \geq \Omega(1)$ where y is x with each bit flipped independently with probability $1/\log(n)$ (Mossel & O’Donnell, 2003). Thus, any variant of the “low-degree algorithm” (Linial et al., 1993) is unlikely to work for this problem.
3. Unlike for the case of all polynomial-size DNF formulas, there are no known reasons for ruling out statistical query (SQ) algorithms for this problem. (The algorithms for learning random DNF formulas cited above can all be couched as SQ algorithms.) Strong SQ lower bounds are known for depth-3 monotone formulas (Feldman et al., 2010), but there are no known strong SQ lower bounds for any subclass of monotone DNF formulas.

Rewards:

A hand shake: Demonstrate a uniform-distribution learning algorithm for Talagrand DNF formulas in the average-case.

A hand shake and a pat on the back: Demonstrate a uniform-distribution learning algorithm for Talagrand DNF formulas in the worst-case.

A surprised look: Prove a super-polynomial strong-SQ lower bound for the class of Talagrand DNF formulas.

References

- Blum, A. (2003). Learning a function of r relevant variables. *Proc. of the 16th Annual Conference on Computational Learning Theory (COLT)* (pp. 731–733). Springer-Verlag.
- Feldman, V., Lee, H. K., & Servedio, R. A. (2010). Lower bounds and hardness amplification for learning shallow monotone formulas. Manuscript.
- Jackson, J. C., Lee, H. K., Servedio, R. A., & Wan, A. (To appear). Learning random monotone DNF. *Discrete Applied Mathematics*. Prelim. ver. in *Proc. of RANDOM'08*.
- Jackson, J. C., & Servedio, R. A. (2006). On learning random DNF formulas under the uniform distribution. *Theory of Computing*, 2, 147–172.
- Linial, N., Mansour, Y., & Nisan, N. (1993). Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, 40, 607–620. Prelim. ver. in *Proc. of FOCS'89*.
- Mossel, E., & O'Donnell, R. (2003). On the noise sensitivity of monotone functions. *Random Structures and Algorithms*, 23, 333–350.
- Sellie, L. (2008). Learning random monotone DNF under the uniform distribution. *Proc. of the 21th Annual Conference on Computational Learning Theory (COLT)* (pp. 181–192).
- Sellie, L. (2009). Exact learning of random dnf over the uniform distribution. *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)* (pp. 45–54).
- Talagrand, M. (1996). How much are increasing sets positively correlated? *Combinatorica*, 16, 243–258.
- Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27, 1134–1142. Prelim. ver. in *Proc. of STOC'84*.

Open Problem: Analyzing Ant Robot Coverage*

Sven Koenig

Computer Science Department
University of Southern California
941 W 37th Street
Los Angeles, CA 90089-0781, USA
skoenig@usc.edu

Abstract

Ant robots can repeatedly and robustly cover terrain by always moving away from the trails that they leave in the terrain. This coverage strategy can be modeled with graph traversal strategies similar to real-time search methods (such as Learning Real-Time A*) and reinforcement learning methods (such as Real-Time Dynamic Programming). The resulting worst-case cover times are known to be exponential in the number of vertices on both directed and undirected graphs in general. The known undirected graphs with large worst-case cover times have unbounded degree vertices. However, existing ant robots navigate on grids, a special case of undirected planar graphs with bounded degree vertices. Their experimental cover times appear to scale almost identically to those of coverage strategies with polynomial worst-case cover times. However, it is an open problem to prove whether the resulting worst-case cover times on grids are indeed polynomial in the number of vertices.

Ant robots are robots that either 1) leave trails in the terrain and use them for navigation, similar to learning graphs by dropping indistinguishable pebbles (Bender et al., 2002), and/or 2) use greedy navigation strategies that depend only on local observations of the terrain and thus require only limited sensing, processing and communication capabilities (Wagner & Bruckstein, 2001). Researchers have built actual ant robots that fit both definitions and cover terrain repeatedly by always moving away from the trails that they leave in the terrain, see Figure 1. Single ant robots (individually) and groups of ant robots (cooperatively) cover terrain robustly even if they do not have any memory, do not know the terrain, cannot maintain maps of the terrain nor plan complete paths. They cover terrain even if some ant robots fail, they are moved without realizing this (say, by people running into them and pushing them accidentally to a different location), the trails are of uneven quality or some trails are destroyed. Their coverage strategy can be modeled with Node Counting (Koenig et al., 2001; Wagner et al., 1999), a graph traversal strategy similar to real-time search methods (such as Learning Real-Time A* (Korf, 1990)) and reinforcement learning methods (such as Real-Time Dynamic Programming (Barto et al., 1995)). Node Counting assigns an integer counter $u(s)$ to every vertex (= node) s of the graph, that represents the amount of trail in that location. All counters are initially zero. Every ant robot always increments the counter of a vertex by one when it enters the vertex and then moves to a successor vertex with the smallest counter (using an arbitrary tie breaking rule), see Figure 3. Thus, it moves to a successor vertex that has been visited the least number of times by ant robots, with the idea to quickly get to a vertex that has not yet been visited. Note that Step 3 of Node Counting is: $u(s) := 1 + u(s)$. For simplicity, we consider only a single ant robot in the following since it is easy to generalize the results to groups of ant robots. Node Counting covers strongly connected graphs repeatedly, which is why we assume in the following that the graphs are strongly connected. The worst-case cover times of Node Counting are known to be exponential in the number of vertices on both directed graphs (trivial proof for the graph topology shown in Figure 4 left) and undirected graphs (longer proof for the graph topology shown in Figure 4 right) in general (Koenig et al., 2001). The known undirected graphs with large worst-case cover times are thus (planar) trees with unbounded degree vertices. However, existing ant robots navigate on grids with blocked and unblocked cells, which are special cases of undirected planar graphs with bounded degree vertices, see Figure 2. The experimental cover times of Node Counting on grids appear to scale almost identically to those of known coverage strategies with polynomial worst-case cover times on all strongly connected graphs. These coverage strategies are similar to Node Counting but more difficult to implement on actual ant robots (Koenig & Simmons, 1992), including Learning Real-Time A*. Step 3 of Learning Real-Time A* is: $u(s) := 1 + \min_{a \in A(s)} u(\text{succ}(s, a))$. We now list interesting open problems for

*This overview of open problems is based on (Svennebring & Koenig, 2003) and (Koenig et al., 2001) and the figures contained therein. It was supported by, or in part by, NSF under contract/grant number 0413196, ARL/ARO under contract/grant number W911NF-08-1-0468 and ONR under contract/grant number N00014-09-1-1031.

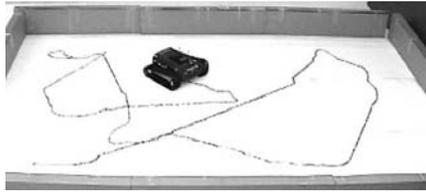


Figure 1: Actual Ant Robot

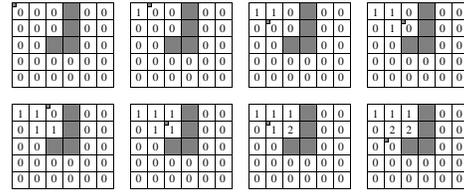


Figure 2: Coverage of Four-Neighbor Grid

We use the following notation: S denotes the finite set of vertices of the graph, and $s_{start} \in S$ denotes the start vertex of an ant robot. $A(s) \neq \emptyset$ is the finite, nonempty set of directed edges that leave vertex $s \in S$. $succ(s, a)$ denotes the successor vertex that results from the traversal of edge $a \in A(s)$ in vertex $s \in S$. We also use two operators with the following semantics: Given a finite set X , the expression “one-of X ” returns an element of X according to an arbitrary rule. A subsequent invocation of “one-of X ” can return the same or a different element. The expression “ $\arg \min_{x \in X} f(x)$ ” returns the elements $x \in X$ that minimize $f(x)$, that is, the set $\{x \in X | f(x) = \min_{x' \in X} f(x')\}$, where f is a function from X to the non-negative integers. Initially, the values $u(s)$ are zero for all $s \in S$.

- Step 1: $s := s_{start}$.
- Step 2: $a := \text{one-of } \arg \min_{a \in A(s)} u(succ(s, a))$.
- Step 3: $u(s) := 1 + u(s)$.
- Step 4: (Traverse edge a .)
- Step 5: $s := succ(s, a)$.
- Step 6: Go to Step 2.

Figure 3: Node Counting

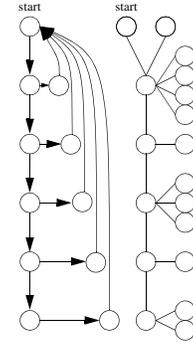


Figure 4: Graphs

Node Counting, the solutions of which would help to lay a solid theoretical foundation for ant robotics and perhaps other kinds of simple agents (such as mobile code that has to explore computer networks): Prove whether the cover times of Node Counting are polynomial in the number of vertices a) for undirected graphs with bounded degree vertices or, if not, b) for grids (a subset of these graphs) if the worst case in both cases is taken over all graphs with a given number of vertices, start vertices and equally good successor vertices (= that is, successor vertices with the smallest counter) and thus tie breaking rules. If not, assume that the ant robot uses the tie breaking rule to select randomly among all equally good successor vertices. Prove whether the resulting cover times are polynomial if the worst case is taken over all graphs with a given number of vertices and start vertices but the average case is taken over all equally good successor vertices. Of course, it is also important to analyze more complex and thus more realistic versions of Node Counting, such as versions that model the saturation of the terrain with trails or the clean-up of trails by the ant robot to avoid such a saturation. For example, Step 3 of a version of Node Counting that models the saturation of the terrain with trails is: with probability $(k - u(s))/k$ execute $u(s) := 1 + u(s)$ for a given positive integer k . Additional information and related work are presented in (Svennebring & Koenig, 2003), in (Koenig et al., 2001) and on the ant robotics web pages at idm-lab.org/antrobots.

References

- Barto, A., Bradtke, S., & Singh, S. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 73, 81–138.
- Bender, M., Fernandez, A., Ron, D., Sahai, A., & Vadhan, S. (2002). The power of a pebble: Exploring and mapping directed graphs. *Information and Computation*, 176, 1–21.
- Koenig, S., & Simmons, R. (1992). *Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains* (Technical Report CMU-CS-93-106). School of Computer Science, Carnegie Mellon University, Pittsburgh (Pennsylvania).
- Koenig, S., Szymanski, B., & Liu, Y. (2001). Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence*, 31, 41–76. The proofs can be found in the technical report ‘The Complexity of Node Counting on Undirected Graphs’ at idm-lab.org/bib/abstracts/Koenig99g.html.
- Korf, R. (1990). Real-time heuristic search. *Artificial Intelligence*, 42, 189–211.
- Svennebring, J., & Koenig, S. (2003). Building terrain-covering ant robots. *Autonomous Robots*, 16, 313–332.
- Wagner, I., & Bruckstein, A. (2001). Special issue on ant robotics. *Annals of Mathematics and Artificial Intelligence*, 31.
- Wagner, I., Lindenbaum, M., & Bruckstein, A. (1999). Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15, 918–933.

On-line variance minimization in $O(n^2)$ per trial?

Elad Hazan
IBM Almaden
650 Harry Road
San Jose, CA 95120
ehazan@cs.princeton.edu

Satyen Kale
Yahoo! Research
4301 Great America Parkway
Santa Clara, CA 95054
skale@yahoo-inc.com

Manfred K. Warmuth*
Department of Computer Science
UC Santa Cruz
manfred@cse.ucsc.edu

Consider the following canonical online learning problem with matrices [WK06]: In each trial t the algorithm chooses a density matrix $\mathbf{W}_t \in \mathcal{R}^{n \times n}$ (i.e., a positive semi-definite matrix with trace one). Then nature chooses a symmetric loss matrix $\mathbf{L}_t \in \mathcal{R}^{n \times n}$ whose eigenvalues lie in the interval $[0, 1]$ and the algorithm incurs loss $\text{tr}(\mathbf{W}_t \mathbf{L}_t)$. The goal is to find algorithms that for any sequence of trials have small regret against the best dyad chosen in hindsight. Here a *dyad* is an outer product $\mathbf{u}\mathbf{u}^\top$ of a unit vector \mathbf{u} in \mathcal{R}^n . More precisely the regret after T trials is defined as follows:

$$\sum_{t=1}^T \text{tr}(\mathbf{W}_t \mathbf{L}_t) - L^*, \quad \text{where } L^* = \inf_{\mathbf{u}: \|\mathbf{u}\|=1} \text{tr}(\mathbf{u}\mathbf{u}^\top \mathbf{L}_{\leq T}) \text{ with } \mathbf{L}_{\leq T} = \sum_{t=1}^T \mathbf{L}_t.$$

Instead of choosing a density matrix \mathbf{W}_t , the algorithm may eigendecompose \mathbf{W}_t as $\sum_i \sigma_i \mathbf{u}_i \mathbf{u}_i^\top$ and choose the eigendyad¹ $\mathbf{u}_i \mathbf{u}_i^\top$ with probability σ_i . If the loss matrix \mathbf{L}_t is a covariance matrix of a random variable, then $\mathbf{u}_i^\top \mathbf{L}_t \mathbf{u}_i$ is the variance in direction \mathbf{u}_i and $\text{tr}(\mathbf{W}_t \mathbf{L}_t)$ the expected variance / loss with respect to \mathbf{W}_t .

Good regret bounds are achieved by a matrix version of the Hedge algorithm [FS97] predicting with:

$$\mathbf{W}_t = \exp(-\eta \mathbf{L}_{<t}) / \text{tr}(\exp(-\eta \mathbf{L}_{<t})),$$

where $\exp(\cdot)$ is the matrix exponential and η a nonnegative learning rate. When η is chosen as $\sqrt{2 \frac{\ln n}{\widehat{L}}}$, where $\widehat{L} \geq L^*$, then the *Matrix Hedge* algorithm achieves a regret bound of $\sqrt{2} \sqrt{\widehat{L} \ln n} + \ln n$ and $\sqrt{2}$ is the best known constant before the leading $\sqrt{\widehat{L} \ln n}$ term.

Note that when the initial matrix \mathbf{W}_1 is the identity matrix and the loss matrices are all diagonal, then Matrix Hedge maintains a distribution over the n unit dyads $\mathbf{e}_i \mathbf{e}_i^\top$ (often called “experts” in this case) and becomes the original Hedge algorithm [FS97] written with diagonal matrices instead of probability and loss vectors. The problem with Matrix Hedge is that it takes $O(n^3)$ time per trial, because the matrix exponential is typically computed by decomposing the matrices and exponentiating the eigenvalues.

Open problem: Is there an $O(n^2)$ per trial algorithm with a regret bound of $O(\sqrt{\widehat{L} \ln n})$?

Why is this a natural problem? Note that for the standard expert setting, the running time of the essentially optimal Hedge algorithm is **linear** in the number of experts n . For the matrix version, the size of all matrices involved is n^2 and we want an $O(n^2)$ per trial algorithm.

An approach based on Follow the Perturbed Leader algorithm. For the original expert setting there is an alternative algorithm to Hedge: Add a vector $\mathbf{r} \in \mathcal{R}^n$ of perturbations to the current total loss $\ell_{<t} \in \mathcal{R}_{>0}^n$ of all n experts and predicts at trial t with the expert $\arg \min_i \ell_{<t,i} + r_i$ of minimum perturbed loss. When r_i is the log of a suitably chosen exponential random variable, then this *Follow the Perturbed Leader* (FPL) algorithm simulates the Hedge algorithm for experts and thus obtains essentially the optimal regret bound [KW05, Kal05].

It is natural to consider matrix versions of FPL for our matrix problem. Now the perturbation is an $n \times n$ matrix \mathbf{R}_t that is added to the loss matrix $\mathbf{L}_{<t}$. Computing a best expert corresponds to finding the minimum eigendyad (i.e. the one corresponding to the minimum eigenvalue) of the perturbed matrix $\mathbf{L}_{<t} + \mathbf{R}_t$, which can be approximately done in $O(n^2)$ time. Thus *Matrix FPL* essentially takes $O(n^2)$ time provided that the perturbation matrix at trial t can be sampled in $O(n^2)$ time. This speedup would be very important because it would open the path for implementing the Matrix Exponentiated Gradient algorithm [TRW05] in $O(n^2)$ time,

*Supported by NSF grant IIS-0917397

¹A dyad $\mathbf{u}\mathbf{u}^\top$, where \mathbf{u} is a unit eigenvector.

bypassing the use of decompositions, and hence have other applications, such as the ones given in [AK07] for efficiently approximating combinatorial problems.

If \mathbf{R}_t is always set to a pre-selected random perturbation matrix \mathbf{R} , and hence does not depend on the current loss matrix $\mathbf{L}_{<t}$, and the adversary is non-adaptive, i.e. the sequence of loss matrices is fixed in advance, then we can allow $O(n^3)$ time or greater for computing \mathbf{R} , because this is only a preprocessing step. In each round a minimum eigendyad can then be approximated in $O(n^2)$ time.

However if $\mathbf{L}_{<t}$ and \mathbf{R}_t do not have a similar eigensystem, then \mathbf{R}_t may not perturb the top eigenvalues of $\mathbf{L}_{<t}$ very much. So for achieving good regret bounds it seems necessary that the perturbation matrix \mathbf{R}_t adapts to $\mathbf{L}_{<t}$. If we allow the algorithm ample $O(n^3)$ time for choosing its perturbation matrix \mathbf{R}_t , then it is trivial to simulate Matrix Hedge with FPL: Simply decompose $\mathbf{L}_{<t}$ in $O(n^3)$ time per trial and then add log exponential perturbations to the eigenvalues as done in [KW05, Kal05] (This corresponds to choosing \mathbf{R}_t to have the same eigenbasis as $\mathbf{L}_{<t}$ with eigenvalues chosen from the log exponential distribution); finally, predict with minimum eigendyad of the perturbed loss matrix. However, this $O(n^3)$ per trial implementation of Matrix FPL is not interesting, because you might as well just use the original Matrix Hedge algorithm that requires the same time.

If we ignore the optimum dependence on the dimension, then by choosing a fixed perturbation with an exponential spectral perturbation and a randomly chosen eigenbasis, we can get an $O(n^2)$ per trial algorithm and $O(n^3)$ preprocessing time. The following theorem can be proved along the same lines as in [HKW10]:

Theorem 1 *For appropriately chosen ε , the expected regret of the algorithm given below is bounded by $O(\sqrt{\widehat{L}r \log n})$, where r is an upper bound on the rank of the loss matrices.*

-
- 1: Sample n real numbers $\sigma_1, \sigma_2, \dots, \sigma_n$ independently from the Laplace distribution with mean 0 and scale $1/\varepsilon$, i.e. the two-sided exponential probability density function $f(x) = \frac{\varepsilon}{2} \exp(-\varepsilon|x|)$.
 - 2: Sample a random orthogonal matrix \mathbf{U} uniformly from the Haar measure.
 - 3: Define $\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top$, where $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$.
 - 4: **for** $t = 1$ to T **do**
 - 5: Let $\mathbf{W}_t = \mathbf{u}_t \mathbf{u}_t^\top$, the minimum eigendyad of the matrix $\mathbf{L}_{<t} + \mathbf{R}$.
 - 6: Predict \mathbf{W}_t and observe the actual loss matrix \mathbf{L}_t . Incur loss $\text{tr}(\mathbf{u}_t \mathbf{u}_t^\top \mathbf{L}_t)$.
 - 7: **end for**
-

Notice that this already resolves the open problem for loss matrices of rank one (or constant rank). This suggests the following direction: The so-called “unit rule” in the usual expert setting says that the worst possible sequence of losses for the experts in a Hedge-type algorithm are the ones where only a single expert incurs loss in each trial. If the analogous statement were true for the FPL-type algorithms suggested above, in the sense that the worst sequence of loss matrices were all rank one, then our open problem would be solved.

Unfortunately, however, it is easy to concoct examples of matrices where for a fixed perturbation matrix, the loss on a rank 2 loss matrix is more than the loss on a sequence of two rank 1 loss matrices. The unit-rule might still be true in an expected sense, but we have been unable to prove such a statement.

References

- [AK07] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *STOC*, pages 227–236, 2007.
- [FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.
- [HKW10] E. Hazan, S. Kale, and M. K. Warmuth. Learning rotations with little regret. In *COLT*, 2010.
- [Kal05] A. Kalai. A perturbation that makes Follow the Leader equivalent to Randomized Weighted Majority. Private communication, December 2005.
- [KW05] D. Kuzmin and M. K. Warmuth. *Optimum Follow the Leader Algorithm*, volume 3559, pages 684–686. Springer Verlag, 2005.
- [TRW05] K. Tsuda, G. Rätsch, and M. K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projections. *Journal of Machine Learning Research*, 6:995–1018, June 2005.
- [WK06] M. K. Warmuth and D. Kuzmin. Online variance minimization. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT '06)*, Pittsburg, June 2006. Springer-Verlag.

Robust Efficient Conditional Probability Estimation

John Langford
Yahoo! Research
jl@yahoo-inc.com

1 The Problem

The problem is finding a general, robust, and efficient mechanism for estimating a conditional probability $P(y|x)$ where robustness and efficiency are measured using techniques from learning reductions.

In particular, suppose we have access to a binary regression oracle B which has two interfaces—one for specifying training information and one for testing. Training information is specified as $B(x', y')$ where x' is an unspecified feature vector and $y' \in [0, 1]$ is a bounded range scalar with no value returned. This operation is stateful, possibly altering the return value of the testing interface in arbitrary ways. Testing is done according to $B(x')$ with a value in $[0, 1]$ returned. The testing operation operation is stateless.

A learning reduction consists of two algorithms R and R^{-1} .

The algorithm R takes as input a single example (x, y) where x is a feature vector and $y \in \{1, \dots, k\}$ is a discrete variable. R then specifies a training example (x', y') for the oracle B . R can then create another training example for B based on all available information. This process repeats some finite number of times before halting without returning information.

A basic observation is that for any oracle algorithm, a distribution $D(x, y)$ over multiclass examples and a reduction R induces a distribution over a sequence $(x', y')^*$ of oracle examples. We collapse this into a distribution $D'(x', y')$ over oracle examples by drawing uniformly from the sequence.

The algorithm R^{-1} takes as input a single example (x, y) and returns a value $v \in [0, 1]$ after using (only) the testing interface of B zero or more times.

We measure the power of an oracle and a reduction according to squared-loss regret according to:

$$\text{reg}(D, R^{-1}) = E_{(x,y) \sim D} [(R^{-1}(x, y) - D(y|x))^2]$$

and similarly letting $\mu_{x'} = E_{(x',y') \sim D'} [y']$.

$$\text{reg}(D', B) = E_{(x',y') \sim D'} (B(x') - \mu_{x'})^2$$

The open problem is to specify R and R^{-1} satisfying the following theorem:

Theorem 1 *For all multiclass distributions $D(x, y)$, for all binary oracles B : The computational complexity of R and R^{-1} are $O(\log k)$ and*

$$\text{reg}(D, R^{-1}) \leq C \text{reg}(D', B)$$

where C is a universal constant.

Alternatively, this open problem is satisfied by proving there exists no deterministic algorithms R, R^{-1} satisfying the above theorem statement.

2 Motivation

The problem of conditional probability estimation is endemic to machine learning applications. In fact, in some branches of machine learning, this is simply considered “the problem”. Typically conditional probability estimation is done in situations where the conditional probability of only one bit is required, however there are a growing number of applications where a well-estimated conditional probability over a more complex object is required. For example, all known methods for solving contextual bandit algorithms over an arbitrary policy class require knowledge of or good estimation of $P(a | x)$ where a is an action.

There is a second intrinsic motivation which is matching the lower bound. No method faster than $O(\log k)$ can be imagined because the label y requires $\log_2 k$ bits to specify and hence read. Similarly it’s easy to prove no learning reduction can provide a regret ratio with $C < 1$.

The motivation for using the learning reduction framework to specify this problem is a combination of generality and the empirical effectiveness in application of learning reductions. Any solution to this will be general because any oracle B can be plugged in, even ones which use many strange kinds of prior information, features, and active multitask hierarchical (insert your favorite adjective here) structure.

3 Related Results

The state of the art is summarized by [1] which shows it's possible to have a learning reduction satisfying the above theorem with either:

1. C replaced by $\log_2^2 k$ (using a binary tree structure)
2. or the computational time increased to $O(k)$ (using an error correcting code structure).

Hence, answering this open problem in the negative shows that there is an inherent computation vs. robustness tradeoff.

There are two other closely related problems, where similar analysis can be done.

1. For multiclass classification, where the goal is predicting the most likely class, a result analogous to the open problem is provable using error correcting tournaments [2].
2. For multiclass classification in a partial label setting, no learning reduction can provide a constant regret guarantee [3].

4 Silly tricks that don't work

Because Learning reductions are not familiar to everyone, we note certain tricks which do not work here to prevent false leads and provide some intuition.

4.1 Ignore B 's predictions and use your favorite learning algorithm instead.

This doesn't work, because the quantification is for all D . Any specified learning algorithm will have some D on which it has nonzero regret. On the other hand, because R calls the oracle at least once, there is a defined induced distribution D' . Since the theorem must hold for all D and B , it must hold for a D your specified learning algorithm fails on and for a B for which $\text{reg}(D', B) = 0$ implying the theorem is not satisfied.

4.2 Feed random examples into B and vacuously satisfy the theorem by making sure that the right hand side is larger than a constant.

This doesn't work because the theorem is stated in terms of squared loss regret rather than squared loss. In particular, if the oracle is given examples of the form (x', y') where $y' \in \{0, 1\}$ is drawn uniformly at random, any oracle specifying $B(x') = 0.5$ has zero regret.

4.3 Feed pseudorandom examples into B and vacuously satisfy the theorem by making sure that the right hand side is larger than a constant.

This doesn't work, because the quantification is "for all binary oracles B ", and there exists one which, knowing the pseudorandom seed, can achieve zero loss (and hence zero regret).

4.4 Just use Boosting to drive the LHS to zero.

Boosting theorems require a stronger oracle—one which provides an edge over some constant baseline for each invocation. The oracle here is not limited in this fashion since it could completely err for a small fraction of invocations.

4.5 Take an existing structure, parameterize it, randomize over the parameterization, and then average over the random elements.

Employing this approach is not straightforward, because the average in D' is over an increased number of oracle examples. Hence, at a fixed expected (over oracle examples) regret, the number of examples allowed to have a large regret is increased.

References

- [1] Alina Beygelzimer, John Langford, Yuri Lifshits, Gregory Sorkin, and Alex Strehl, Conditional Probability Tree Estimation Analysis and Algorithms, UAI 2009.
- [2] Alina Beygelzimer, John Langford, and Pradeep Ravikumar, Error-Correcting Tournaments, ALT 2009.
- [3] Alina Beygelzimer and John Langford, The Offset Tree for Learning with Partial Labels, KDD 2009.

Can We Learn to Gamble Efficiently?

Jacob Abernethy
UC Berkeley
jake@eecs.berkeley.edu

1 Introduction

Betting is an important problem faced by millions of sports fans each day. Presented with an upcoming matchup between team A and team B, and given the opportunity to place a 50/50 wager on either, where should a gambler put her money? This decision is not, of course, made in isolation: both teams will have played a number of decided matches with other teams throughout the season. Furthermore, a reasonable assumption to make is that the relation “A tends to beat B” is transitive. Under transitivity, the best prediction strategy is clearly to sort the teams by their abilities and predict according to this *ranking*.

The obvious difficulty is that the best ranking of the teams is not known an advance. But there’s a more subtle issue: even with knowledge of all match outcomes in advance, i.e. a list of items of the form (team $X <$ team Y), it’s NP-hard to determine the best ranking of the teams when the outcomes are noisy. This is exactly the infamous Minimum Feedback Arc Set problem.

The question we pose is as follows: can we design a non-trivial online prediction strategy in this setting which achieves *vanishing regret* relative to the best ranking in hindsight, even when the latter is computationally infeasible?

It is tempting to believe this is impossible, as competing with the best ranking would appear tantamount to finding the best ranking. However, this assertion is false: the algorithm *need not* learn a ranking explicitly, it must only output a prediction ($X < Y$) or ($Y < X$) when presented with a pair (X, Y) , and these predictions do not necessarily have to satisfy transitivity. Indeed, consider the following simple algorithm: treat each team pair (X, Y) as an independent learning problem (ignoring all other matchups). In the long run, this will achieve vanishing regret with respect to the best ranking. So why is this not desirable? The trivial approach unfortunately admits a bad regret bound: the algorithm must see $O(n)$ matches per team before it can start to make decent predictions. On the other hand, there is an information-theoretic approach that requires only $O(\log n)$ observations per team—the downside, unfortunately, is that this requires exponential computation. We would like to achieve this rate with an efficient method.

2 Problem Setup

We have a set of n teams with indices $i = 1, 2, \dots, n$. A learner is presented with a sequence of pairs (i_t, j_t) for $t = 1, \dots, T$ and must predict $\hat{y}_t \in [-1, 1]$, where $\hat{y}_t = 1$ implies that the learner believes that team i_t will beat team j_t , and vice versa when $\hat{y}_t = -1$. After making her prediction, the learner observes the outcome $y_t \in \{-1, 1\}$ and suffers loss $\ell(\hat{y}_t, y_t) := (1 - \hat{y}_t y_t)/2$.

An online prediction algorithm A is a function that outputs predictions \hat{y}_t given input of the data $(i_1, j_1, y_1), \dots, (i_{t-1}, j_{t-1}, y_{t-1})$ and the current matchup (i_t, j_t) . We can compare such a prediction algorithm to any *offline comparator* class \mathcal{F} , which is any collection of “skew-symmetric” mappings $\phi : [n] \times [n] \rightarrow \{-1, 1\}$, namely those that satisfy $\phi(i, j) = -\phi(j, i)$ for all i, j (required since if i beats j then j doesn’t beat i). We’ll consider two such classes, the class \mathcal{F}_{all} of *all* such mappings, and the class $\mathcal{F}_{\text{perm}}$ of *permutations*, i.e. those mappings ϕ which satisfy the transitive property, $\phi(i, j) = 1$ and $\phi(j, k) = 1$ implies $\phi(i, k) = 1$. The *regret* of any algorithm A with respect to any \mathcal{F} is defined as

$$\text{Regret}_{\mathcal{F}}(A) := \sum_{t=1}^T \ell(\hat{y}_t, y_t) - \min_{\phi \in \mathcal{F}} \sum_{t=1}^T \ell(\phi(i_t, j_t), y_t)$$

Lemma 1 For any algorithm A , $\text{Regret}_{\mathcal{F}_{\text{perm}}}(A) \leq \text{Regret}_{\mathcal{F}_{\text{all}}}(A)$.

Proof:[sketch] This follows trivially from the fact that $\mathcal{F}_{\text{perm}} \subset \mathcal{F}_{\text{all}}$ ■

Lemma 2 There exists an inefficient algorithm A such that $\text{Regret}_{\mathcal{F}_{\text{perm}}}(A) = O(\sqrt{Tn \log n})$.

Proof:[sketch] If we treat each permutation $\phi \in \mathcal{F}_{\text{perm}}$ as an “expert” and run a standard experts algorithm, then well-known results (see, e.g., Cesa-Bianchi and Lugosi [1]) imply that the regret will scale as $O(\sqrt{T \log(\text{no. of experts})})$. The result follows since there are $n!$ permutations, and $\log n! = \Theta(n \log n)$. ■

Lemma 3 There exists an efficient algorithm A such that $\text{Regret}_{\mathcal{F}_{\text{all}}}(A) = O(\sqrt{Tn^2})$.

Proof:[sketch] Let i and j be arbitrary and assume without loss of generality that $i < j$. If we treat the problem “does i beat j ?” as an independent learning problem, then we can imagine that we have two experts: “ i wins” and “ j wins”. So if the matchup (i, j) occurs $T_{i,j}$ times throughout the season, then on only these particular events we are guaranteed to achieve $O(\sqrt{T_{i,j}})$ regret using a standard experts algorithm. If we do this for every pair i, j independently, then we can achieve

$$\text{Regret}_{\mathcal{F}_{\text{all}}} = O\left(\sum_{(i,j):i<j} \sqrt{T_{i,j}}\right).$$

Using the fact that, for any $z_1, \dots, z_m \geq 0$,

$$\sum_{j=1}^m \sqrt{z_j} \leq \sqrt{m} \sqrt{\sum_{j=1}^m z_j}$$

and that $\sum_{(i,j):i<j} T_{i,j} = T$, immediately gives the desired bound. ■

This final Lemma suggests that sub-linear regret is not difficult for learning rankings. Unfortunately, this bound scales quite poorly in n . This is precisely because this algorithm, when predicting the outcome of a pair (X, Y) , does not leverage the information from any other matchups. For example, when the biggest winner plays the biggest loser for the first time, this algorithm would guess the outcome is a tossup.

Open Problem: Is it possible to find an efficient algorithm that closes the gap between the $O(\sqrt{Tn^2})$ bound and the information-theoretic $O(\sqrt{Tn \log n})$ rate? Is it even possible to do any better than $O(\sqrt{Tn^2})$ efficiently?

Prior Work

While not stated as we do here, essentially the same open question was posed by Kleinberg et al [2] in 2008. Their work also provides a very useful background on learning to rank.

Acknowledgments

I would like to thanks to Adam Kalai for originally suggesting this open question, and to Hamid Nazer Zadeh and Eli Ben-Sasson for joining our unsuccessful attempt at solving it. Adam gets additional credit for regularly finding counterexamples to nearly all of our proposed solutions. Great work, Adam, really.

References

- [1] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge Univ Pr, 2006.
- [2] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *ACM COLT*, 2008.

Active Learning on Trees and Graphs

Nicolò Cesa-Bianchi
Università degli Studi di Milano
Italy
nicolo.cesa-bianchi@unimi.it

Claudio Gentile
Università dell’Insubria
Italy
claudio.gentile@uninsubria.it

Fabio Vitale **Giovanni Zappella**
Università degli Studi di Milano
Italy
fabio.vitale@unimi.it
giovanni.zappella@studenti.unimi.it

Abstract

We investigate the problem of active learning on a given tree whose nodes are assigned binary labels in an adversarial way. Inspired by recent results by Guillory and Bilmes, we characterize (up to constant factors) the optimal placement of queries so to minimize the mistakes made on the non-queried nodes. Our query selection algorithm is extremely efficient, and the optimal number of mistakes on the non-queried nodes is achieved by a simple and efficient mincut classifier. Through a simple modification of the query selection algorithm we also show optimality (up to constant factors) with respect to the trade-off between number of queries and number of mistakes on non-queried nodes. By using spanning trees, our algorithms can be efficiently applied to general graphs, although the problem of finding optimal and efficient active learning algorithms for general graphs remains open. Towards this end, we provide a lower bound on the number of mistakes made on arbitrary graphs by any active learning algorithm using a number of queries which is up to a constant fraction of the graph size.

1 Introduction

The abundance of networked data in various application domains (web, social networks, bioinformatics, etc.) motivates the development of scalable and accurate graph-based prediction algorithms. An important topic in this area is the graph binary classification problem: Given a graph with unknown binary labels on its nodes, the learner receives the labels on a subset of the nodes (the training set) and must predict the labels on the remaining vertices. This is typically done by relying on some notion of label regularity depending on the graph topology, such as that nearby nodes are likely to be labeled similarly. Standard approaches to this problem predict with the assignment of labels minimizing the induced cutsize (e.g., [4, 5]), or by binarizing the assignment that minimizes certain real-valued extensions of the cutsize function (e.g., [10, 2, 3] and references therein).

In the active learning version of this problem the learner is allowed to choose the subset of training nodes. Similarly to standard feature-based learning, one expects active methods to provide a significant boost of predictive ability compared to a noninformed (e.g., random) draw of the training set. The following simple example provides some intuition of why this could happen when the labels are chosen by an adversary, which is the setting considered in this paper. Consider a “binary star system” of two star-shaped graphs whose centers are connected by a bridge, where one star is a constant fraction bigger than the other. The adversary draws two random binary labels and assigns the first label to all nodes of the first star graph, and the second label to all nodes of the second star graph. Assume that the training set size is two. If we choose the centers of the two stars and predict with a mincut strategy,¹ we are guaranteed to make zero mistakes on all unseen vertices. On the other hand, if we query two nodes at random, then with constant probability both of them will belong to the bigger star, and all the unseen labels of the smaller star will be mistaken. This simple example shows that the gap between the performance of passive and active learning on graphs can be made arbitrarily big.

In general, one would like to devise a strategy for placing a certain budget of queries on the vertices of a given graph. This should be done so as to minimize the number of mistakes made on the non-queried nodes by some reasonable classifier like mincut. This question has been investigated from a theoretical viewpoint

¹A mincut strategy considers all labelings consistent with the labels observed so far, and chooses among them one that minimizes the resulting cutsize over the whole graph.

by Guillory and Bilmes [6], and by Afshani et al. [1]. Our work is related to an elegant result from [6] which bounds the number of mistakes made by the mincut classifier on the worst-case assignment of labels in terms of $\Phi/\Psi(L)$. Here Φ is the cutsize induced by the unknown labeling, and $\Psi(L)$ is a function of the query (or training) set L , which depends on the structural properties of the (unlabeled) graph. For instance, in the above example of the binary system, the value of $\Psi(L)$ when the query set L includes just the two centers is 1. This implies that for the binary system graph, Guillory and Bilmes' bound on the mincut strategy is Φ mistakes in the worst case (note that in the above example $\Phi \leq 1$). Since $\Psi(L)$ can be efficiently computed on any given graph and query set L , the learner's task might be reduced to finding a query set L that maximizes $\Psi(L)$ given a certain query budget (size of L). Unfortunately, no feasible general algorithm for solving this maximization problem is known, and so one must resort to heuristic methods — see [6].

In this work we investigate the active learning problem on graphs in the important special case of trees. We exhibit a simple iterative algorithm which, combined with a mincut classifier, is optimal (up to constant factors) on any given labeled tree. This holds even if the algorithm is not given information on the actual cutsize Φ . Our method is extremely efficient, requiring $\mathcal{O}(n \ln Q)$ time for placing Q queries in an n -node tree, and space linear in n . As a byproduct of our analysis, we show that Ψ can be efficiently maximized over trees to within constant factors. Hence the bound $\min_L \Phi/\Psi(L)$ can be achieved efficiently.

Another interesting question is what kind of trade-off between queries and mistakes can be achieved if the learner is not constrained by a given query budget. We show that a simple modification of our selection algorithm is able to trade-off queries and mistakes in an optimal way up to constant factors.

Finally, we prove a general lower bound for predicting the labels of any given graph (not necessarily a tree) when the query set is up to a constant fraction of the number of vertices. Our lower bound establishes that the number of mistakes must then be at least a constant fraction of the cutsize weighted by the effective resistances. This lower bound apparently yields a contradiction to the results of Afshani et al. [1], who constructs the query set adaptively. This apparent contradiction is also obtained via a simple counterexample that we detail in Section 5.

2 Preliminaries and basic notation

A labeled tree (T, \mathbf{y}) is a tree $T = (V, E)$ whose nodes $V = \{1, \dots, n\}$ are assigned binary labels $\mathbf{y} = (y_1, \dots, y_n) \in \{-1, +1\}^n$. We measure the label regularity of (T, \mathbf{y}) by the *cutsize* $\Phi_T(\mathbf{y})$ induced by \mathbf{y} on T , i.e., $\Phi_T(\mathbf{y}) = |\{(i, j) \in E : y_i \neq y_j\}|$. We consider the following *active learning* protocol: given a tree T with unknown labeling \mathbf{y} , the learner obtains all labels in a *query set* $L \subseteq V$, and is then required to predict the labels of the remaining nodes $V \setminus L$. Active learning algorithms work in two-phases: a *selection* phase, where a query set of given size is constructed, and a *prediction* phase, where the algorithm receives the labels of the query set and predicts the labels of the remaining nodes. Note that the only labels ever observed by the algorithm are those in the query set. In particular, no labels are revealed during the prediction phase.

We measure the ability of the algorithm by the number of prediction mistakes made on $V \setminus L$, where it is reasonable to expect this number to depend on both the unknown cutsize $\Phi_T(\mathbf{y})$ and the number $|L|$ of requested labels. A slightly different prediction measure is considered in Section 4.3.

Given a tree T and a query set $L \subseteq V$, a node $i \in V \setminus L$ is a **fork node generated by L** if and only if there exist three distinct nodes $i_1, i_2, i_3 \in L$ that are connected to i through edge disjoint paths. Let $\text{FORK}(L)$ be the set of all fork nodes generated by L . Then L^+ is the query set obtained by adding to L all the generated fork nodes, i.e., $L^+ \triangleq L \cup \text{FORK}(L)$. We say that $L \subseteq V$ is **0-forked** iff $L^+ \equiv L$. Note that L^+ is 0-forked. That is, $\text{FORK}(L^+) \equiv \emptyset$ for all $L \subseteq V$.

Given a node subset $S \subseteq V$, we use $T \setminus S$ to denote the forest obtained by removing from the tree T all nodes in S and all edges incident to them. Moreover, given a second tree T' , we denote by $T \setminus T'$ the forest $T \setminus V'$, where V' is the set of nodes of T' . Given a query set $L \subseteq V$, a **hinge-tree** is any connected component of $T \setminus L^+$. We call **connection node** of a hinge-tree a node of L adjacent to any node of the hinge tree. We distinguish between 1-hinge and 2-hinge trees. A **1-hinge-tree** has one connection node only, whereas a **2-hinge-tree** has two (note that a hinge tree cannot have more than two connection nodes because L^+ is zero-forked, see Figure 1).

3 The active learning algorithm

We now describe the two phases of our active learning algorithm. For the sake of exposition, we call **SEL** the selection phase and **PRED** the prediction phase. **SEL** returns a 0-forked query set $L_{\text{SEL}}^+ \subseteq V$ of desired size. **PRED** takes in input the query set L_{SEL}^+ and the set of labels y_i for all $i \in L_{\text{SEL}}^+$. Then **PRED** returns a prediction for the labels of all remaining nodes $V \setminus L_{\text{SEL}}^+$.

In order to see the way **SEL** operates, we formally introduce the function Ψ^* . This is the reciprocal of the Ψ function introduced in [6] and mentioned in Section 1.

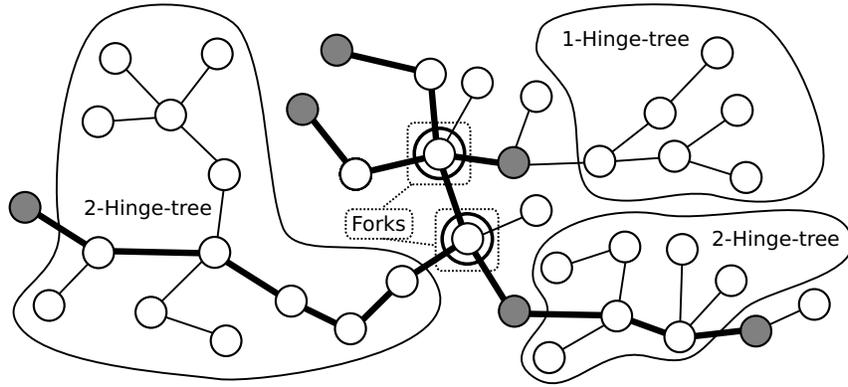


Figure 1: A tree $T = (V, E)$ whose nodes are shaded (the query set L) or white (the set $V \setminus L$). The shaded nodes are also the connection nodes of the depicted hinge trees (not all hinge trees are contoured). The fork nodes generated by L are denoted by double circles. The thick black edges connect the nodes in L .

Definition 1 Given a tree $T = (V, E)$ and a set of nodes $L \subseteq V$,

$$\Psi^*(L) \triangleq \max_{\emptyset \neq V' \subseteq V \setminus L} \frac{|V'|}{|\{(i, j) \in E : i \in V', j \in V \setminus V'\}|}.$$

In words, $\Psi^*(L)$ measures the largest set of nodes not in L that share the least number of edges with nodes in L . From the adversary's viewpoint, $\Psi^*(L)$ can be described as the largest return in mistakes per unit of cutsize invested. We now move on to the description of the algorithms SEL and PRED.

The **selection algorithm** SEL greedily computes a query set that minimizes Ψ^* to within constant factors. To this end, SEL exploits Lemma 10 (a) (see Section 4.2) stating that, for any fixed query set L , the subset $V' \subseteq V$ maximizing $\frac{|V'|}{|\{(i, j) \in E : i \in V', j \in V \setminus V'\}|}$ is always included in a connected component of $T \setminus L$. Thus

SEL places its queries in order to end up with a query set L_{SEL}^+ such that the largest component of $T \setminus L_{\text{SEL}}^+$ is as small as possible.

SEL operates as follows. Let $L_t \subseteq L$ be the set including the first t nodes chosen by SEL, T_{max}^t be the largest connected component of $T \setminus L_{t-1}$, and $\sigma(T', i)$ be the size (number of nodes) of the largest component of the forest $T' \setminus \{i\}$, where T' is any tree. At each step $t = 1, 2, \dots$, SEL simply picks the node $i_t \in T_{\text{max}}^t$ that minimizes $\sigma(T_{\text{max}}^t, i)$ over i and sets $L_t = L_{t-1} \cup \{i_t\}$. During this iterative construction, SEL also maintains a set containing all fork nodes generated in each step by adding nodes i_t to the sets L_{t-1} .² After the desired number of queries is reached (also counting the queries that would be caused by the stored fork nodes), SEL has terminated the construction of the query set L_{SEL} . The final query set L_{SEL}^+ , obtained by adding all stored fork nodes to L_{SEL} , is then returned.

The **Prediction Algorithm** PRED receives in input the labeled nodes of the 0-forked query set L_{SEL}^+ and computes a mincut assignment. Since each component of $T \setminus L_{\text{SEL}}^+$ is either a 1-hinge-tree or a 2-hinge-tree, PRED is simple to describe and is also very efficient. The algorithm predicts all the nodes of hinge-tree \mathcal{T} using the same label $\hat{y}_{\mathcal{T}}$. This label is chosen according to the following two cases:

1. If \mathcal{T} is a 1-hinge-tree, then $\hat{y}_{\mathcal{T}}$ is set to the label of its unique connection node;
2. If \mathcal{T} is a 2-hinge-tree and the labels of its two connection nodes are equal, then $\hat{y}_{\mathcal{T}}$ is set to the label of its connection nodes, otherwise $\hat{y}_{\mathcal{T}}$ is set as the label of the closer connection node (ties are broken arbitrarily).

In Section 6 we show that SEL requires overall $\mathcal{O}(|V| \log Q)$ time and $\mathcal{O}(|V|)$ memory space for selecting Q query nodes. Also, we will see that the total running time taken by PRED for predicting all nodes in $V \setminus L$ is linear in $|V|$.

4 Analysis

For a given tree T , we denote by $m_A(L, \mathbf{y})$ the number of prediction mistakes that algorithm A makes on the labeled tree (T, \mathbf{y}) when given the query set L . Introduce the function

$$m_A(L, K) = \max_{\mathbf{y} : \Phi_T(\mathbf{y}) \leq K} m_A(L, \mathbf{y})$$

²In Section 6 we will see that during each step $L_{t-1} \rightarrow L_t$ at most a single new fork node may be generated.

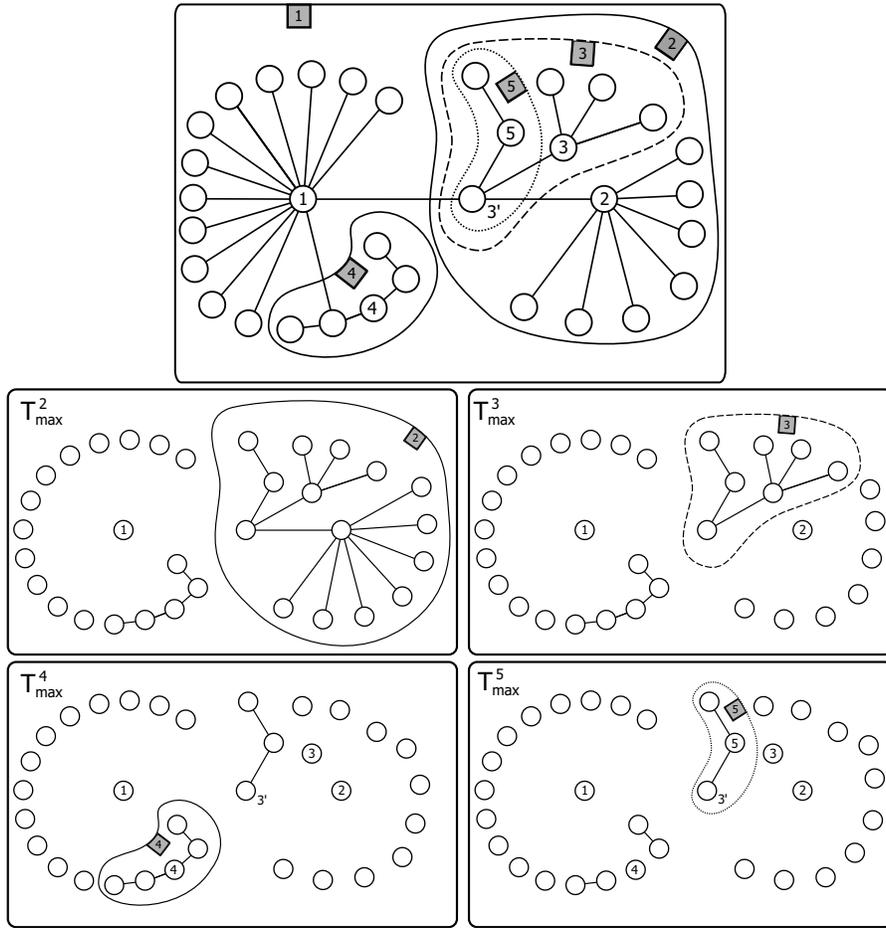


Figure 2: The SEL algorithm at work. The upper pane shows the initial tree $T = T_{\max}^1$ (in the box tagged with “1”), and the subsequent subtrees T_{\max}^2 , T_{\max}^3 , T_{\max}^4 , and T_{\max}^5 . The left pane also shows the nodes selected by SEL in chronological order. The four lower panes show the connected components of $T \setminus L_t$ resulting from this selection. Observe that at the end of round 3, SEL detects the generation of fork node $3'$. This node gets stored, and is added to L_{SEL} at the end of the selection process.

denoting the number of prediction mistakes made by A with query set L on all labeled trees with cutsizes bounded by K . We will also find it useful to deal with the “lower bound” function $\text{LB}(L, K)$. This is the maximum expected number of mistakes that any prediction algorithm A can be forced to make on the labeled tree (T, \mathbf{y}) when the query set is L and the cutsizes is not larger than K .

We show that the number of mistakes made by PRED on any labeled tree when using the query set L_{SEL}^+ satisfies

$$m_{\text{PRED}}(L_{\text{SEL}}^+, K) \leq 10 \text{LB}(L, K)$$

for all query sets $L \subseteq V$ of size up to $\frac{1}{8}|L_{\text{SEL}}^+|$. Though neither SEL nor PRED do know the actual cutsizes of the labeled tree (T, \mathbf{y}) , the combined use of these procedures is competitive against any algorithm that knows the cutsizes budget K beforehand.

While this result implies the optimality (up to constant factors) of our algorithm, it does not relate the mistake bound to the cutsizes, which is a clearly interpretable measure of the label regularity. In order to address this issue, we show that our algorithm also satisfies the bound

$$m_{\text{PRED}}(L_{\text{SEL}}^+, \mathbf{y}) \leq 4 \Psi^*(L) \Phi_T(\mathbf{y})$$

for all query sets $L \subseteq V$ of size up to $\frac{1}{8}|L_{\text{SEL}}^+|$. The proof of these results needs a number of preliminary lemmas.

Lemma 2 For any tree $T = (V, E)$ it holds that $\min_{v \in V} \sigma(T, v) \leq \frac{1}{2}|V|$.

Proof: Let $i \in \text{argmin}_{v \in V} \sigma(T, v)$. For the sake of contradiction, assume there exists a component $T_i = (V_i, E_i)$ of $T \setminus \{i\}$ such that $|V_i| > |V|/2$. Let s be the sum of the sizes all other components. Since

$|V_i| + s = |V| - 1$, we know that $s \leq |V|/2 - 1$. Now let j be the node adjacent to i which belongs to V_i and $T_j = (V_j, E_j)$ be the largest component of $T \setminus \{j\}$. There are only two cases to consider: either $V_j \subset V_i$ or $V_j \cap V_i \equiv \emptyset$. In the first case, $|V_j| < |V_i|$. In the second case, $V_j \subseteq \{i\} \cup (T \setminus V_i)$, which implies $|V_j| \leq 1 + s \leq |V|/2 < |V_i|$. In both cases, $i \notin \operatorname{argmin}_{v \in V} \sigma(T, v)$, which provides the desired contradiction. \blacksquare

Lemma 3 For all subsets $L \subset V$ of the nodes of a tree $T = (V, E)$ we have $|L^+| \leq 2|L|$.

Proof: Pick an arbitrary node of T and perform a depth-first visit of all nodes in T . This visit induces an ordering $\mathcal{T}_1, \mathcal{T}_2, \dots$ of the connected components in $T \setminus L$ based on the order of the nodes visited first in each component. Now let $\mathcal{T}'_1, \mathcal{T}'_2, \dots$ be such that each \mathcal{T}'_i is a component of \mathcal{T}_i extended to include all nodes of L adjacent to nodes in \mathcal{T}_i . Then the ordering implies that, for $i \geq 2$, \mathcal{T}'_i shares exactly one node (which must be a leaf) with all previously visited trees. Since in any tree the number of nodes of degree larger than two must be strictly smaller than the number of leaves, we have $|\operatorname{FORK}(\mathcal{T}'_i)| < |\Lambda_i|$ where, with slight abuse of notation, we denote by $\operatorname{FORK}(\mathcal{T}'_i)$ the set of all fork nodes in subtree \mathcal{T}'_i . Also, we let Λ_i be the set of leaves of \mathcal{T}'_i . This implies that, for $i = 1, 2, \dots$, each fork node in $\operatorname{FORK}(\mathcal{T}'_i)$ can be injectively associated with one of the $|\Lambda_i| - 1$ leaves of \mathcal{T}'_i that are not shared with any of the previously visited trees. Since $|\operatorname{FORK}(L)|$ is equal to the sum of $|\operatorname{FORK}(\mathcal{T}'_i)|$ over all indices i , this implies that $|\operatorname{FORK}(L)| \leq |L|$. \blacksquare

Lemma 4 Let $L_{t-1} \subseteq L_{\operatorname{SEL}}$ be the set of the first $t - 1$ nodes chosen by SEL. Given any tree $T = (V, E)$, the largest subtree of $T \setminus L_{t-1}$ contains no more than $\frac{2}{t}|V|$ nodes.

Proof: Recall that i_s denotes the s -th node selected by SEL during the incremental construction of the query set L_{SEL} , and that T_{\max}^s is the largest component of $T \setminus L_{s-1}$. The first t steps of the recursive splitting procedure performed by SEL can be associated with a splitting tree T' defined in the following way. The internal nodes of T' are T_{\max}^s , for $s \geq 1$. The children of T_{\max}^s are the connected components of $T_{\max}^s \setminus \{i_s\}$, i.e., the subtrees of T_{\max}^s created by the selection of i_s . Hence, each leaf of T' is bijectively associated with a tree in $T \setminus L_t$.

Let T'_{no1} be the tree obtained from T' by deleting all leaves. Each node of T'_{no1} is one of the t subtrees split by SEL during the construction of L_t . As T_{\max}^t is split by i_t , it is a leaf in T'_{no1} . We now add a second child to each internal node s of T'_{no1} having a single child. This second child of s is obtained by merging all the subtrees belonging to leaves of T' that are also children of s . Let T'' be the resulting tree.

We now compare the cardinality of T_{\max}^t to that of the subtrees associated with the leaves of T'' . Let Λ be the set of all leaves of T'' and $\Lambda_{\operatorname{add}} = T'' \setminus T'_{\operatorname{no1}} \subset \Lambda$ be the set of all leaves added to T'_{no1} to obtain T'' . First of all, note that $|T_{\max}^t|$ is not larger than the number of nodes in any leaf of T'_{no1} . This is because the selection rule of SEL ensures that T_{\max}^t cannot be larger than any subtree associated with a leaf in T'_{no1} , since it contains no node selected before time t . In what follows, we write $|s|$ to denote the size of the forest or subtree associated with a node s of T'' . We now prove the following claim:

Claim. For all $\ell \in \Lambda$, $|T_{\max}^t| \leq |\ell|$, and for all $\ell \in \Lambda_{\operatorname{add}}$, $|T_{\max}^t| - 1 \leq |\ell|$.

Proof of Claim. The first part just follows from the observation that any $\ell \in \Lambda$ was split by SEL before time t . In order to prove the second part, pick a leaf $\ell \in \Lambda_{\operatorname{add}}$. Let ℓ' be its unique sibling in T'' and let p be the parent of ℓ and ℓ' , also in T'' . Lemma 2 applied to the subtree p implies $|\ell'| \leq \frac{1}{2}|p|$. Moreover, since $|\ell| + |\ell'| = |p| - 1$, we obtain $|\ell| + 1 \geq \frac{1}{2}|p| \geq |\ell'| \geq |T_{\max}^t|$, the last inequality using the first part of the claim. This implies $|T_{\max}^t| - 1 \leq |\ell|$, and the claim is proven.

Let now $N(\Lambda)$ be the number of nodes in subtrees and forests associated with the leaves of T'' . With each internal node of T'' we can associate a node of L_{SEL} which does not belong to any leaf in Λ . Moreover, the number $|T'' \setminus \Lambda|$ of internal nodes in T'' is bigger than the number $|\Lambda_{\operatorname{add}}|$ of internal nodes of T'_{no1} to which a child has been added. Since these subtrees and forests are all distinct, we obtain $N(\Lambda) + |T'' \setminus \Lambda| < N(\Lambda) + |\Lambda_{\operatorname{add}}| \leq |V|$. Hence, using the above claim we can write $N(\Lambda) \geq (|\Lambda| - |\Lambda_{\operatorname{add}}|)|T_{\max}^t| + |\Lambda_{\operatorname{add}}|(|T_{\max}^t| - 1)$, which implies $|T_{\max}^t| \leq (N(\Lambda) + |\Lambda_{\operatorname{add}}|)/|\Lambda| \leq |V|/|\Lambda|$. Since each internal node of T'' has at least two children, we have that $|\Lambda| \geq |T''|/2 \geq |T'_{\operatorname{no1}}|/2 = t/2$. Hence, we can conclude that $|T_{\max}^t| \leq 2|V|/t$. \blacksquare

4.1 Lower bounds

We now state and prove a lower bound on the number of mistakes that any prediction algorithm (even knowing the cutsizes budget K) makes on any given tree, when the query set L is 0-forked. The bound depends on the following quantity: Given a tree $T(V, E)$, a node subset $L \subseteq V$ and an integer K , the **component function** $\Upsilon(L, K)$ is the sum of the sizes of the K largest components of $T \setminus L$, or $|V \setminus L|$ if $T \setminus L$ has less than K components.

Theorem 5 For all trees $T = (V, E)$, for all 0-forked subsets $L^+ \subseteq V$, and for all cutsizes budgets $K = 0, 1, \dots, |V| - 1$, we have that $\text{LB}(L^+, K) \geq \frac{1}{2}\Upsilon(L^+, K)$.

Proof: We describe an adversarial strategy causing any algorithm to make at least $\Upsilon(L^+, K)/2$ mistakes even when the cutsize budget K is known beforehand. Since L^+ is 0-forked, each component of $T \setminus L^+$ is a hinge-tree. Let F_{\max} be the set of the K largest hinge-trees of $T \setminus L^+$, and $E(T)$ be the set of all edges in E incident to at least one node of a hinge-tree \mathcal{T} . The adversary creates at most one ϕ -edge³ in each edge set $E(\mathcal{T}_1)$ for all 1-hinge-trees $\mathcal{T}_1 \in F_{\max}$, exactly one ϕ -edge in each edge set $E(\mathcal{T}_2)$ for all 2-hinge-trees $\mathcal{T}_2 \in F_{\max}$, and no ϕ -edges in the edge set $E(T)$ of any remaining hinge-tree $\mathcal{T} \notin F_{\max}$. This is done as follows. By performing a depth-first visit of T , the adversary can always assign disagreeing labels to the two connection nodes of each 2-hinge-tree in F_{\max} , and agreeing labels to the two connection nodes of each 2-hinge-tree not in F_{\max} . Then, for each hinge-tree $\mathcal{T} \in F_{\max}$, the adversary assigns a unique random label to all nodes of \mathcal{T} , forcing $|\mathcal{T}|/2$ mistakes in expectation. The labels of the remaining hinge-trees not in F_{\max} are chosen in agreement with their connection nodes. \blacksquare

Remark 1 Note that Theorem 5 holds for all query sets, not only those that are 0-forked, since any adversarial strategy for a query set L^+ can force at least the same mistakes on the subset $L \subseteq L^+$. Note also that it is not difficult to modify the adversarial strategy described in the proof of Theorem 5 in order to deal with algorithms that are allowed to adaptively choose the query nodes in L depending on the labels of the previously selected nodes. The adversary simply assigns the same label to each node in the query set and then forces, with the same method described in the proof, $\frac{1}{2}\Upsilon(L^+, \frac{K}{2})$ mistakes in expectation on the $\frac{K}{2}$ largest hinge-trees. Thus there are at most two ϕ -edges in each edge set $E(T)$ for all hinge-trees \mathcal{T} , yielding at most K ϕ -edges in total. The resulting (slightly weaker) bound is $\text{LB}(L^+, K) \geq \frac{1}{2}\Upsilon(L^+, \frac{K}{2})$. Theorem 8 and Corollary 9 can also be easily rewritten in order to extend the results in this direction.

4.2 Upper bounds

We now bound the total number of mistakes that PRED makes on any labeled tree when the queries are decided by SEL. We use Lemma 2 and 3, together with the two lemmas below, to prove that $m_{\text{PRED}}(L_{\text{SEL}}^+, K) \leq 10 \text{LB}(L, K)$ for all cutsize budgets K and for all node subset $L \subseteq V$ such that $|L| \leq \frac{1}{8}|L_{\text{SEL}}^+|$.

Lemma 6 For all labeled trees (T, \mathbf{y}) and for all 0-forked query sets $L^+ \subseteq V$, the number of mistakes made by PRED satisfies $m_{\text{PRED}}(L^+, \mathbf{y}) \leq \Upsilon(L^+, \Phi_T(\mathbf{y}))$.

Proof: As in the proof of Theorem 5, we first observe that each component of $T \setminus L^+$ is a hinge-tree. Let $E(T)$ be the set of all edges in E incident to nodes of a hinge-tree \mathcal{T} , and F_ϕ be the set of hinge-trees such that, for all $\mathcal{T} \in F_\phi$, at least one edge of $E(\mathcal{T})$ is a ϕ -edge. Since $E(\mathcal{T}) \cap E(\mathcal{T}') \equiv \emptyset$ for all $\mathcal{T}, \mathcal{T}' \in T \setminus L^+$, we have that $|F_\phi| \leq \Phi_T(\mathbf{y})$. Moreover, since for any $\mathcal{T} \notin F_\phi$ there are no ϕ -edges in $E(\mathcal{T})$, the nodes of \mathcal{T} must be labeled as its connections nodes. This, together with the prediction rule of PRED, implies that PRED makes no mistakes over any of the hinge-trees $\mathcal{T} \notin F_\phi$. Hence, the number of mistakes made by PRED is bounded by the sum of the sizes of all hinge-trees $\mathcal{T} \in F_\phi$, which (by definition of Υ) is bounded by $\Upsilon(L^+, \Phi_T(\mathbf{y}))$. \blacksquare

The next lemma, whose proof is a bit involved, provides the relevant properties of the component function $\Upsilon(\cdot, \cdot)$. Figure 3 helps visualizing the main ingredients of the proof.

Lemma 7 Given a tree $T = (V, E)$, for all node subsets $L \subseteq V$ such that $|L| \leq \frac{1}{2}|L_{\text{SEL}}|$ and for all integers k , we have: (a) $\Upsilon(L_{\text{SEL}}, k) \leq 5\Upsilon(L, k)$; (b) $\Upsilon(L_{\text{SEL}}, 1) \leq \Upsilon(L, 1)$.

Proof: We prove part (a) by constructing, via SEL, three bijective mappings $\mu_1, \mu_2, \mu_3 : \mathcal{P}_{\text{SEL}} \rightarrow \mathcal{P}_L$, where \mathcal{P}_{SEL} is a suitable partition of $T \setminus L_{\text{SEL}}$, \mathcal{P}_L is a subset of 2^V such that any $S \in \mathcal{P}_L$ is all contained in a single connected component of $T \setminus L$, and the union of the domains of the three mappings covers the whole set $T \setminus L_{\text{SEL}}$. The mappings μ_1, μ_2 and μ_3 are shown to satisfy, for all forests⁴ $F \in \mathcal{P}_{\text{SEL}}$,

$$|F| \leq |\mu_1(F)|, \quad |F| \leq 2|\mu_2(F)|, \quad |F| \leq 2|\mu_3(F)|.$$

Since each $S \in \mathcal{P}_L$ is all contained in a connected component of $T \setminus L$, this we will enable us to conclude that, for each tree $T' \in T \setminus L$, the forest of all trees $T \setminus L_{\text{SEL}}$ mapped (via any of these mappings) to any node subset of T' has at most five times the number of nodes of T' . This would prove the statement in (a).

³A ϕ -edge (i, j) is one where $y_i \neq y_j$.

⁴In this proof, $|\mu(A)|$ denotes the number of nodes in the set (of nodes) $\mu(A)$. Also, with a slight abuse of notation, for all forests $F \in \mathcal{P}_{\text{SEL}}$, we denote by $|F|$ the sum of the number of nodes in all trees of F . Finally, whenever $F \in \mathcal{P}_{\text{SEL}}$ contains a single tree, we refer to F as it were a tree, rather than a (singleton) forest containing only one tree.

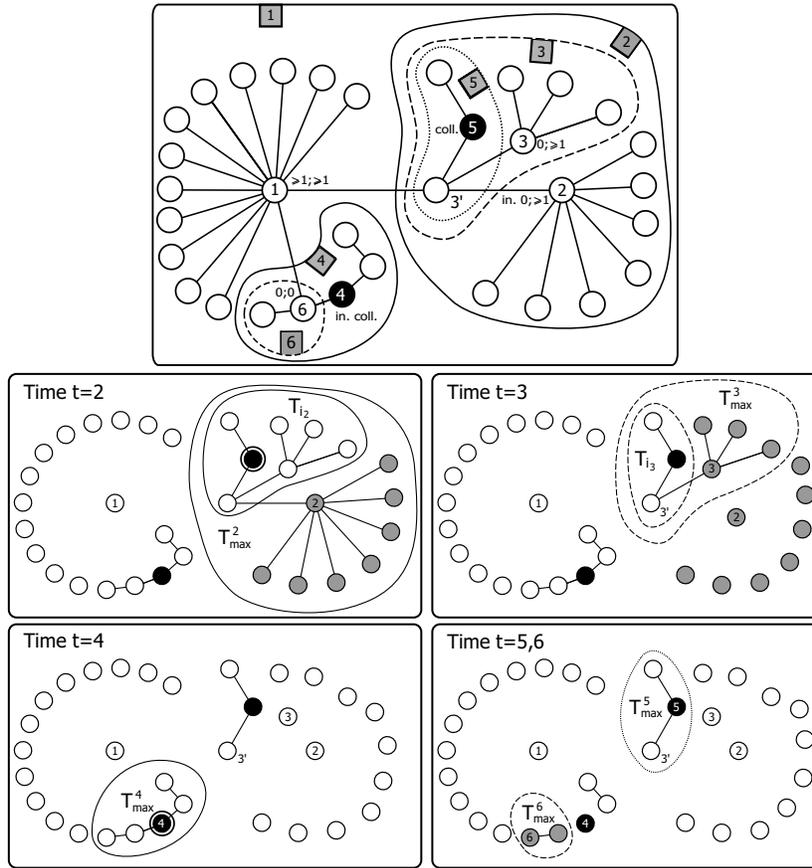


Figure 3: The upper pane illustrates the different kinds of nodes chosen by SEL. Numbers in the square tags indicate the first six subtrees T_{\max}^t , and their associated nodes i_t , selected by SEL. Node i_1 is a $[\geq 1; \geq 1]$ -node, i_2 is an initial $[0; \geq 1]$ -node, i_3 is a (noninitial) $[0; \geq 1]$ -node, i_4 is an initial collision node, i_5 is a (noninitial) collision node, and i_6 is a $[0; 0]$ -node. As in Figure 2, we denote by $3'$ the fork node generated by the inclusion of i_3 into L_{SEL} . Note that node i_6 may be chosen arbitrarily among the four nodes in $T_{\max}^4 \setminus i_4$. The two black nodes are the set of nodes we are competing against, i.e., the nodes in the query set L . Forest $T \setminus L$ is made up of one large subtree and two small subtrees. In the lower panes we illustrate some steps of the proof of Lemma 7, with reference to the upper pane. **Time $t=2$:** Trees T_{\max}^2 and T_{i_2} are shown. As explained in the proof, $|T_{i_2}| \leq |T_{\max}^2 \setminus T_{i_2}|$. The circled black node is captured by i_2 . The nodes of tree $T_{\max}^2 \setminus T_{i_2}$ are shaded, and can be used for mapping any ζ -component through μ_2 . **Time $t=3$:** Trees T_{\max}^3 and T_{i_3} are shown. Again, one can easily verify that $|T_{i_3}| \leq |T_{\max}^3 \setminus T_{i_3}|$. As before, the nodes of $T_{\max}^3 \setminus T_{i_3}$ are shaded, and can be used for mapping any ζ -component via μ_2 . The reader can see that, according to the injectivity of μ_2 , these grey nodes are well separated from the ones in $T_{\max}^2 \setminus T_{i_2}$. **Time $t=4$:** T_{\max}^4 and the initial collision node i_4 are depicted. The latter is enclosed in a circled black node since it captures itself. **Time $t=5, 6$:** We depicted trees T_{\max}^5 and T_{\max}^6 , together with nodes i_5 and i_6 . Node i_5 is a collision node, which is not initial since it was already captured by the $[0; \geq 1]$ -node i_2 . Node i_6 is a $[0; 0]$ node, so that the whole tree T_{\max}^6 is completely included in a component (the largest, in this case) of $T \setminus L$. Tree T_{\max}^6 can be used for mapping via μ_3 any ζ -component. The resulting forest $T \setminus L_6$ includes several single-node trees and one two-node tree. If i_6 is the last node selected by L_{SEL} , then each component of $T \setminus L_6$ can be exploited by mapping μ_1 , since in this specific case none of these components contains nodes of L , i.e., there are no ζ -components left.

The construction of these mappings requires some auxiliary definitions. We call ζ -component each connected component of $T \setminus L_{\text{SEL}}$ containing at least one node of L . Let i_t be the t -th node selected by SEL during the incremental construction of the query set L_{SEL} . We distinguish between four kinds of nodes chosen by SEL — see Figure 3 for an example.

Node i_t is:

1. A *collision node* if it belongs to $L_{\text{SEL}} \cap L$;
2. a $[0; 0]$ -node if, at time t , the tree T_{max}^t does not contain any node of L ;
3. a $[0; \geq 1]$ -node if, at time t , the tree T_{max}^t contains $k \geq 1$ nodes $j_1, \dots, j_k \in L$ all belonging to the same connected component of $T_{\text{max}}^t \setminus \{i_t\}$;
4. a $[\geq 1; \geq 1]$ -node if $i_t \notin L$ and, at time t , the tree T_{max}^t contains $k \geq 2$ nodes $j_1, \dots, j_k \in L$, which do not belong to the same connected component of $T_{\text{max}}^t \setminus \{i_t\}$.

We now turn to building the three mappings.

μ_1 simply maps each tree $T' \in T \setminus L_{\text{SEL}}$ that is *not* a ζ -component to the node set of T' itself. This immediately implies $|F| \leq |\mu_1(F)|$ for all forests F (which are actually single trees) in the domain of μ_1 . Mappings μ_2 and μ_3 deal with the ζ -components of $T \setminus L_{\text{SEL}}$. Let Z be the set of all such ζ -components, and denote by $V_{0;0}$, $V_{0;1}$, and $V_{1;1}$ the set of all $[0; 0]$ -nodes, $[0; \geq 1]$ -nodes, and $[\geq 1; \geq 1]$ -nodes, respectively. Observe that $|V_{1;1}| < |L|$. Combined with the assumption $|L_{\text{SEL}}| \geq 2|L|$, this implies that $|V_{0;0}| + |V_{0;1}|$ plus the total number of collision nodes must be larger than $|L|$; as a consequence, $|V_{0;0}| + |V_{0;1}| > |Z|$. Each node $i_t \in V_{0;1}$ chosen by SEL splits the tree T_{max}^t into one component T_{i_t} containing at least one node of L and one or more components all contained in a single tree T'_{i_t} of $T \setminus L$. Now mapping μ_2 can be constructed incrementally in the following way. For each $[0; \geq 1]$ -node selected by SEL at time t , μ_2 sequentially maps any ζ -component generated to the set of nodes in $T_{\text{max}}^t \setminus T_{i_t}$, the latter being just a subset of a component of $T \setminus L$. A future time step $t' > t$ might feature the selection of a new $[0; \geq 1]$ -node within T_{i_t} , but mapping μ_2 would cover a different subset of such component of $T \setminus L$. Now, applying Lemma 2 to tree T_{max}^t , we can see that $|T_{\text{max}}^t \setminus T_{i_t}| \geq |T_{\text{max}}^t|/2$. Since the selection rule of SEL guarantees that the number of nodes in T_{max}^t is larger than the number of nodes of any ζ -component, we have $|F| \leq 2|\mu_2(F)|$, for any ζ -component F considered in the construction of μ_2 .

Mapping μ_3 maps all the remaining ζ -components that are not mapped through μ_2 . Let \sim be an equivalence relation over $V_{0;0}$ defined as follows: $i \sim j$ iff i is connected to j by a path containing only $[0; 0]$ -nodes and nodes in $V \setminus (L_{\text{SEL}} \cup L)$. Let $i_{t_1}, i_{t_2}, \dots, i_{t_k}$ be the sequence of nodes of any given equivalence class $[C]_{\sim}$, sorted according to SEL's chronological selection. Lemma 4 applied to tree $T_{\text{max}}^{t_1}$ shows that $|T_{\text{max}}^{t_k}| \leq 2|T_{\text{max}}^{t_1}|/k$. Moreover, the selection rule of SEL guarantees that the number of nodes of $T_{\text{max}}^{t_k}$ cannot be smaller than the number of nodes of any ζ -component. Hence, for each equivalence class $[C]_{\sim}$ containing k nodes of type $[0; 0]$, we map through μ_3 a set F_C of k arbitrarily chosen ζ -components to $T_{\text{max}}^{t_1}$. Since the size of each ζ -component is $\leq |T_{\text{max}}^{t_k}|$, we can write $|F_C| \leq k|T_{\text{max}}^{t_k}| \leq 2|T_{\text{max}}^{t_1}|$, which implies $|F_C| \leq 2|\mu_3(F_C)|$ for all F_C in the domain of μ_3 . Finally, observe that the number of ζ -components that are not mapped through μ_2 cannot be larger than $|V_{0;0}|$, thus the union of mappings μ_2 and μ_3 do actually map all ζ -components. This, in turn, implies that the union of the domains of the three mappings covers the whole set $T \setminus L_{\text{SEL}}$, thereby concluding the proof of part (a).

The proof of (b) is built on the definition of collision nodes, $[0; 0]$ -nodes, $[0; \geq 1]$ -nodes and $[\geq 1; \geq 1]$ -nodes given in part (a). Let $L_t \subseteq L_{\text{SEL}}$ be the set of the first t nodes chosen by SEL. Here, we make a further distinction within the collision and $[0; \geq 1]$ -nodes. We say that during the selection of node $i_t \in V_{0;1}$, the nodes in $L \cap T_{\text{max}}^t$ are *captured* by i_t . This notion of capture extends to collision nodes by saying that a collision node $i_t \in L \cap L_{\text{SEL}}$ just *captures itself*. We say that i_t is an *initial* $[0; \geq 1]$ -node (resp., *initial* collision node) if i_t is a $[0; \geq 1]$ -node (resp., collision node) such that the whole set of nodes in L captured by i_t contains no nodes captured so far. See Figure 3 for reference. The simple observation leading to the proof of part (b) is the following. If i_t is a $[0; 0]$ -node, then T_{max}^t cannot be larger than the component of $T \setminus L$ that contains T_{max}^t , which in turn cannot be larger than $\Upsilon(L, 1)$. This would already imply $\Upsilon(L_{t-1}, 1) \leq \Upsilon(L, 1)$. Let now i_t be an initial $[0; \geq 1]$ -node and T_{i_t} be the unique component of $T_{\text{max}}^t \setminus \{i_t\}$ containing one or more nodes of L . Applying Lemma 2 to tree T_{max}^t we can see that $|T_{i_t}|$ cannot be larger than $|T_{\text{max}}^t \setminus T_{i_t}|$, which in turn cannot be larger than $\Upsilon(L, 1)$. If at time $t' > t$ the procedure SEL selects $i_{t'} \in T_{i_t}$ then $|T_{\text{max}}^{t'}| \leq |T_{i_t}| \leq \Upsilon(L, 1)$. Hence, the maximum integer q such that $\Upsilon(L_q, 1) > \Upsilon(L, 1)$ is bounded by the number of $[\geq 1; \geq 1]$ -nodes plus the number of initial $[0; \geq 1]$ -nodes plus the number of initial collision nodes. We now bound this sum as follows. The number of $[\geq 1; \geq 1]$ -nodes is clearly bounded by $|L| - 1$. Also, any initial $[0; \geq 1]$ -node or initial collision node selected by SEL captures at least a new node in L , thereby implying that the total number of initial $[0; \geq 1]$ -node or initial collision node must be $\leq |L|$. After $q = 2|L| - 1$ rounds, we are sure that the size of the largest tree of T_{max}^q is not larger than the size of the largest component of $T \setminus L$, i.e., $\Upsilon(L, 1)$. \blacksquare

We now put the above lemmas together to prove our main result concerning the number of mistakes made by PRED on the query set chosen by SEL.

Theorem 8 For all trees T and all outsize budgets K , the number of mistakes made by PRED on the query set L_{SEL}^+ satisfies

$$m_{\text{PRED}}(L_{\text{SEL}}^+, K) \leq \min_{L \subseteq V : |L| \leq \frac{1}{8}|L_{\text{SEL}}^+|} 10 \text{LB}(L, K).$$

Proof: Pick any $L \subseteq V$ such that $|L| \leq \frac{1}{8}|L_{\text{SEL}}^+|$. Then

$$m_{\text{PRED}}(L_{\text{SEL}}^+, K) \stackrel{(\text{Lem. 6})}{\leq} \Upsilon(L_{\text{SEL}}^+, K) \stackrel{(\text{A})}{\leq} \Upsilon(L_{\text{SEL}}, K) \stackrel{(\text{Lem. 7 (a)})}{\leq} 5\Upsilon(L^+, K) \stackrel{(\text{Thm. 5})}{\leq} 10 \text{LB}(L^+, K) \stackrel{(\text{B})}{\leq} 10 \text{LB}(L, K).$$

Inequality (A) holds because $L_{\text{SEL}} \subseteq L_{\text{SEL}}^+$, and thus $T \setminus L_{\text{SEL}}^+$ has connected components of smaller size than L_{SEL} . In order to apply Lemma 7 (a), we need the condition $|L^+| \leq \frac{1}{2}|L_{\text{SEL}}|$. This condition is seen to hold after combining Lemma 3 with our assumptions: $|L^+| \leq 2|L| \leq \frac{1}{4}|L_{\text{SEL}}^+| \leq \frac{1}{2}|L_{\text{SEL}}|$. Finally, inequality (B) holds because any adversarial strategy using query set L can also be used with the larger query set $L^+ \supseteq L$. ■

Note also that Theorem 5 and Lemma 6 imply the following statement about the optimality of PRED over 0-forked query sets.

Corollary 9 For all trees T , for all outsize budgets K , and for all 0-forked query sets $L^+ \subseteq V$, the number of mistakes made by PRED satisfies $m_{\text{PRED}}(L^+, K) \leq 2\text{LB}(L^+, K)$.

In the rest of this section we derive a more interpretable bound on $m_{\text{PRED}}(L^+, \mathbf{y})$ based on the function Ψ^* introduced in [6]. To this end, we prove that L_{SEL} minimizes Ψ^* up to constant factors, and thus is an optimal query set according to the analysis of [6].

For any subset $V' \subseteq V$, let $\Gamma(V', V \setminus V')$ be the number of edges between nodes of V' and nodes of $V \setminus V'$. Using this notation, we can write

$$\Psi^*(L) = \max_{\emptyset \neq V' \subseteq V \setminus L} \frac{|V'|}{\Gamma(V', V \setminus V')}.$$

Lemma 10 For any tree $T = (V, E)$ and any $L \subseteq V$ the following holds.

- (a) A maximizer of $\frac{|V'|}{\Gamma(V', V \setminus V')}$ exists which is included in the node set of a single component of $T \setminus L$;
- (b) $\Psi^*(L) \leq \Upsilon(L, 1)$.

Proof: Let V'_{max} be any maximizer of $\frac{|V'|}{\Gamma(V', V \setminus V')}$. For the sake of contradiction, assume that the nodes of V'_{max} belong to $k \geq 2$ components $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_k \in T \setminus L$. Let $V'_i \subset V'_{\text{max}}$ be the subset of nodes included in the node set of \mathcal{T}_i , for $i = 1, \dots, k$. Then $|V'| = \sum_{i \leq k} |V'_i|$ and $\Gamma(V', V \setminus V') = \sum_{i \leq k} \Gamma(V'_i, V \setminus V'_i)$. Now let $i^* = \arg\max_{i \leq k} |V'_i|/\Gamma(V'_i, V \setminus V'_i)$. Since $(\sum_i a_i)/(\sum_i b_i) \leq \max_i a_i/b_i$ for all $a_i, b_i \geq 0$, we immediately obtain $\Psi(V'_{i^*}) \geq \Psi(V'_{\text{max}})$, contradicting our assumption. This proves (a). Part (b) is an immediate consequence of (a). ■

Lemma 11 For any tree $T = (V, E)$ and any 0-forked subset $L^+ \subseteq V$ we have $\Upsilon(L^+, 1) \leq 2\Psi^*(L^+)$.

Proof: Let \mathcal{T}_{max} be the largest component of $T \setminus L^+$ and V_{max} be its node set. Since L^+ is a 0-forked query set, \mathcal{T}_{max} must be either a 1-hinge-tree or a 2-hinge-tree. Since the only edges that connect a hinge-tree to external nodes are the edges leading to connection nodes, we find that $\Gamma(V_{\text{max}}, V \setminus V_{\text{max}}) \leq 2$. We can now write

$$\Psi^*(L^+) = \max_{\emptyset \neq V' \subseteq V \setminus L^+} \frac{|V'|}{\Gamma(V', V \setminus V')} \geq \frac{|V_{\text{max}}|}{\Gamma(V_{\text{max}}, V \setminus V_{\text{max}})} \geq \frac{|V_{\text{max}}|}{2} = \frac{\Upsilon(L^+, 1)}{2}$$

thereby concluding the proof. ■

Lemma 12 For any tree $T = (V, E)$ and any subset $L \subseteq V$ we have $\Psi^*(L^+) \leq \Psi^*(L)$.

Proof: Let V'_{max} be any set maximizing $\Psi^*(L^+)$. Since $V'_{\text{max}} \in V \setminus L^+$, V'_{max} cannot contain any node of $L \subseteq L^+$. Hence

$$\Psi^*(L) = \max_{\emptyset \neq V' \subseteq V \setminus L} \frac{|V'|}{\Gamma(V', V \setminus V')} \geq \frac{|V'_{\text{max}}|}{\Gamma(V'_{\text{max}}, V \setminus V'_{\text{max}})} = \Psi^*(L^+)$$

which concludes the proof. ■

We now put together the previous lemmas to show that the query set L_{SEL} minimizes Ψ^* up to constant factors.

Theorem 13 For any tree $T = (V, E)$ we have $\Psi^*(L_{\text{SEL}}) \leq \min_{L \subseteq V : |L| \leq \frac{1}{4}|L_{\text{SEL}}|} 2\Psi^*(L)$.

Proof: Let L be a query set such that $|L| \leq |L_{\text{SEL}}|/4$. Then we have the following chain of inequalities:

$$\Psi^*(L_{\text{SEL}}) \stackrel{\text{(Lemma 10 (b))}}{\leq} \Upsilon(L_{\text{SEL}}, 1) \stackrel{\text{(Lemma 7 (b))}}{\leq} \Upsilon(L^+, 1) \stackrel{\text{(Lemma 11)}}{\leq} 2\Psi^*(L^+) \stackrel{\text{(Lemma 12)}}{\leq} 2\Psi^*(L).$$

In order to apply Lemma 7 (b), we need the condition $|L^+| \leq \frac{1}{2}|L_{\text{SEL}}|$. This condition holds because, by Lemma 3, $|L^+| \leq 2|L| \leq \frac{1}{2}|L_{\text{SEL}}|$. \blacksquare

Finally, as promised, the following corollary contains an interpretable mistake bound for PRED run with a query set returned by SEL.

Corollary 14 For any labeled tree (T, \mathbf{y}) , the number of mistakes made by PRED when run with query set L_{SEL}^+ satisfies

$$m_{\text{PRED}}(L_{\text{SEL}}^+, \mathbf{y}) \leq 4 \min_{L \subseteq V : |L| \leq \frac{1}{8}|L_{\text{SEL}}^+|} \Psi^*(L) \Phi_T(\mathbf{y}).$$

Proof: Observe that PRED assigns labels to nodes in $V \setminus L_{\text{SEL}}^+$ so as to minimize the resulting cutsize given the labels in the query set L_{SEL}^+ . We can then invoke [6, Lemma 1], which bounds the number of mistakes made by the mincut strategy in terms of the functions Ψ^* and the cutsize. This yields

$$m_{\text{PRED}}(L_{\text{SEL}}^+, \mathbf{y}) \stackrel{[6, \text{Lemma 1}]}{\leq} 2\Psi^*(L_{\text{SEL}}^+) \Phi_T(\mathbf{y}) \stackrel{\text{(A)}}{\leq} 2\Psi^*(L_{\text{SEL}}) \Phi_T(\mathbf{y}) \stackrel{\text{(Theorem 13)}}{\leq} 4\Psi^*(L) \Phi_T(\mathbf{y}).$$

Inequality (A) holds because $L_{\text{SEL}} \subseteq L_{\text{SEL}}^+$, and thus $T \setminus L_{\text{SEL}}^+$ has connected components of smaller size than L_{SEL} . In order to apply Theorem 13, we need the condition $|L| \leq \frac{1}{4}|L_{\text{SEL}}|$, which follows from a simple combination of Lemma 3 and our assumptions: $|L| \leq \frac{1}{8}|L_{\text{SEL}}^+| \leq \frac{1}{4}|L_{\text{SEL}}|$. \blacksquare

Remark 2 A mincut algorithm exists which efficiently predicts even when the query set L is not 0-forked (thereby gaining a factor of 2 in the cardinality of the competing query sets L – see Theorem 8 and Corollary 14). This algorithm is a “batch” variant of the TreeOpt algorithm analyzed in [7]. The algorithm can be implemented in such a way that the total time for predicting $|V| - |L|$ labels is $\mathcal{O}(|V|)$.

4.3 Automatic calibration of the number of queries

A key aspect to the query selection task is deciding when to stop asking queries. Since the more queries are asked the less mistakes are made afterwards, a reasonable way to deal with this trade-off is to minimize the number of queries issued during the selection phase plus the number of mistakes made during the prediction phase. For a given pair $A = \langle S, P \rangle$ of prediction and selection algorithms, we denote by $[q + m]_A$ the sum of queries made by S and prediction mistakes made by P . Similarly to m_A introduced in Section 4, $[q + m]_A$ has to scale with the cutsize $\Phi_T(\mathbf{y})$ of the labeled tree (T, \mathbf{y}) under consideration.

As a simple example of computing $[q + m]_A$, consider a line graph $T = (V, E)$. Since each query set on T is 0-forked, Theorem 5 and Corollary 9 ensure that an optimal strategy for selecting the queries in T is choosing a sequence of nodes such that the distance between any pair of neighbor nodes in L is equal. The total number of mistakes that can be forced on $V \setminus L$ is, up to a constant factor, $(|V|/|L|)\Phi_T(\mathbf{y})$. Hence, the optimal value of $[q + m]_A$ is about

$$|L| + \frac{|V|}{|L|} \Phi_T(\mathbf{y}). \quad (1)$$

Minimizing the above expression over $|L|$ clearly requires knowledge of $\Phi_T(\mathbf{y})$, which is typically unavailable. In this section we investigate a method for choosing the number of queries when the labeling is known to be sufficiently regular, that is when a bound K is known on the cutsize $\Phi_T(\mathbf{y})$ induced by the adversarial labeling.⁵

We now show that when a bound K on the cutsize is known, a simple modification of SEL (we call it SEL \star) exists which optimizes the $[q + m]_A$ criterion. This means that the combination of SEL \star and PRED can trade-off optimally (up to constant factors) queries against mistakes.

⁵In [1] a labeling \mathbf{y} of a graph G is said to be α -balanced if, after the elimination of all ϕ -edges, each connected component of G is not smaller than $\alpha|V|$ for some known constant $\alpha \in (0, 1)$. In the case of labeled trees, the α -balancing condition is stronger than our regularity assumption. This is because any α -balanced labeling \mathbf{y} implies $\Phi_T(\mathbf{y}) \leq 1/\alpha - 1$. In fact, getting back to the line graph example, we immediately see that, if \mathbf{y} is α -balanced, then the optimal number of queries $|L|$ is order of $\sqrt{|V|(1/\alpha - 1)}$, which is also $\inf_A [q + m]_A$.

Given a selection algorithm S and a prediction algorithm P , define $[q + m]_{(S,P)}$ by

$$[q + m]_{(S,P)} = \min_{Q \geq 1} (Q + m_P(L_{S(Q)}, K))$$

where $L_{S(Q)}$ is the query set output by S given query budget Q , and $m_P(L_{S(Q)}, K)$ is the maximum number of mistakes made by P with query set $L_{S(Q)}$ on any labeling \mathbf{y} with $\Phi_T(\mathbf{y}) \leq K$ — see definition in Section 4. Define also $[q + m]_{\text{OPT}} = \inf_{S,P} [q + m]_{(S,P)}$, where $\text{OPT} = \langle S^*, P^* \rangle$ is an optimal pair of selection and prediction algorithms. If SEL knows the size of the query set L^* selected by S^* , so that SEL can choose a query budget $Q = 8|L^*|$, then a direct application of Theorem 8 guarantees that $|L_{\text{SEL}}^+| + m_{\text{PRED}}(L_{\text{SEL}}^+, K) \leq 10 [q + m]_{\text{OPT}}$. We now show that SEL^* , the announced modification of SEL, can efficiently search for a query set size Q such that $Q + m_{\text{PRED}}(L_{\text{SEL}^*}^+(Q), K) = \mathcal{O}([q + m]_{\text{OPT}})$ when only K , rather than $|L^*|$, is known. In fact, Theorem 5 and Corollary 9 ensure that $m_{\text{PRED}}(L_{\text{SEL}}^+, K) = \Theta(\Upsilon(L_{\text{SEL}}^+, K))$. When K is given as side information, SEL^* can operate as follows. For each $t \leq |V|$, the algorithm builds the query set L_t^+ and computes $\Upsilon(L_t^+, K)$. Then it finds the smallest value t^* minimizing $t + \Upsilon(L_t^+, K)$ over all $t \leq |V|$, and selects $L_{\text{SEL}^*} \equiv L_{t^*}$. We stress that the above is only possible because the algorithm can estimate within constant factors its own future mistake bound (Theorem 5 and Corollary 9), and because the combination of SEL and PRED is competitive against all query sets whose size is a constant fraction of $|L_{\text{SEL}}^+|$ — see Theorem 8. Putting together, we have shown the following result.

Theorem 15 *For all trees (T, \mathbf{y}) , for all cutsize budgets K , and for all labelings \mathbf{y} such that $\Phi_T(\mathbf{y}) \leq K$, the combination of SEL^* and PRED achieves $|L_{\text{SEL}^*}| + m_{\text{PRED}}(L_{\text{SEL}^*}, K) = \mathcal{O}([q + m]_{\text{OPT}})$ when K is given to SEL^* as input.*

Just to give a few simple examples of how SEL^* works, consider a star graph. It is not difficult to see that in this case $t^* = 1$ independent of K , i.e., SEL^* always selects the center of the star, which is intuitively the optimal choice. If T is the “binary system” mentioned in the introduction, then $t^* = 2$ and SEL^* always selects the centers of the two stars, again independent of K . At the other extreme, if T is a line graph, then SEL^* picks the query nodes in such a way that the distance between two consecutive nodes of L in T is (up to a constant factor) equal to $\sqrt{|V|/K}$. Hence $|L| = \Theta(\sqrt{|V|K})$, which is the minimum of (1) over $|L|$ when $\Phi_T(\mathbf{y}) \leq K$.

5 On the prediction of general graphs

In this section we provide a general lower bound for prediction on arbitrary labeled graphs (G, \mathbf{y}) . We then contrast this lower bound to some results contained in Afshani et al. [1].

Let $\Phi_G^R(\mathbf{y})$ be the sum of the effective resistances (see, e.g., [9]) on the ϕ -edges of $G = (V, E)$. The theorem below shows that any prediction algorithm using any query set L such that $|L| \leq \frac{1}{4}|V|$ makes at least order of $\Phi_G^R(\mathbf{y})$ mistakes. This lower bound holds even if the algorithm is allowed to use a randomized adaptive strategy for choosing the query set L , that is, a randomized strategy where the next node of the query set is chosen after receiving the labels of all previously chosen nodes.

Theorem 16 *Given a labeled graph (G, \mathbf{y}) , for all $K \leq |V|/2$, there exists a randomized labeling strategy such that for all prediction algorithms A choosing a query set of size $|L| \leq \frac{1}{4}|V|$ via a possibly randomized adaptive strategy, the expected number of mistakes made by A on the remaining nodes $V \setminus L$ is at least $K/4$, while $\Phi_G^R(\mathbf{y}) \leq K$.*

The above lower bound (whose proof is omitted) appears to contradict an argument by Afshani et al. [1, Section 5]. This argument establishes that for any $\varepsilon > 0$ there exists a randomized algorithm using at most $K \ln(3/\varepsilon) + K \ln(|V|/K) + \mathcal{O}(K)$ queries on any given graph $G = (V, E)$ with cutsize K , and making at most $\varepsilon|V|$ mistakes on the remaining vertices. This contradiction is easily obtained through the following simple counterexample: assume G is a line graph where all node labels are +1 but for $K = o(|V|/\ln|V|)$ randomly chosen nodes, which are also given random labels. For all $\varepsilon = o(\frac{K}{|V|})$, the above argument implies that order of $K \ln|V| = o(|V|)$ queries are sufficient to make at most $\varepsilon|V| = o(K)$ mistakes on the remaining nodes, among which $\Omega(K)$ have random labels — which is clearly impossible.

6 Efficient Implementation

In this section we describe an efficient implementation of SEL and PRED. We will show that the total time needed for selecting Q queries is $\mathcal{O}(|V| \log Q)$, the total time for predicting $|V| - Q$ nodes is $\mathcal{O}(|V|)$, and that the overall memory space is again $\mathcal{O}(|V|)$.

In order to locate the largest subtree of $T \setminus L_{t-1}$, the algorithm maintains a priority deque [8] D containing at most Q items. This data-structure enables to find and eliminate the item with the smallest (resp., largest) key in time $\mathcal{O}(1)$ (resp., time $\mathcal{O}(\log Q)$). In addition, the insertion of a new element takes time $\mathcal{O}(\log Q)$.

Each item in D has two records: a reference to a node in T and the priority key associated with that node. Just before the selection of the⁶ t -th query node i_t , the Q references point to nodes contained in the Q largest subtrees in $T \setminus L_{t-1}$, while the corresponding keys are the sizes of such subtrees. Hence at time t the item *top* of D having the largest key points to a node in T_{\max}^t .

First, during an initialization step, SEL creates, for each edge $(i, j) \in E$, a directed edge $[i, j]$ from i to j and the twin directed edge $[j, i]$ from j to i . During the construction of L_{SEL} the algorithm also stores and maintains the current size $\sigma(D)$ of D , i.e., the total number of items contained in D . We first describe the way SEL finds node i_t in T_{\max}^t . Then we will see how SEL can efficiently augment the query set L_{SEL} to obtain L_{SEL}^+ .

Starting from the node r of T_{\max}^t referred to by⁷ D , SEL performs a depth-first visit of T_{\max}^t , followed by the elimination of the item with the largest key in D . For the sake of simplicity, consider T_{\max}^t as rooted at node r . Given any edge (i, j) , we let T_i and T_j be the two subtrees obtained from T_{\max}^t after removing edge (i, j) , where T_i contains node i , and T_j contains node j . During each backtracking step of the depth-first visit from a node i to a node j , SEL stores the number of nodes $|T_i|$ contained in T_i . This number gets associated with $[j, i]$. Observe that this task can be accomplished very efficiently, since $|T_i|$ is equal to 1 plus the number of nodes of the union of $T_{c(i)}$ over all children $c(i)$ of i . These numbers can be recursively calculated by summing the size values that SEL associates with all direct edges $[i, c(i)]$ in the previous backtracking steps. Just after storing the value $|T_i|$, the algorithm also stores $|T_j| = |T_{\max}^t| - |T_i|$ and associates this value with the twin directed edge $[i, j]$. The size of T_{\max}^t is then stored in D as the key record of the pointer to node r .

It is now important to observe that the quantity $\sigma(T_{\max}^t, i)$ used by SEL (see Section 3) is simply the largest key associated with the directed edges $[i, j]$ over all j such that (i, j) is an edge of T_{\max}^t . Hence, a new depth-first visit is enough to find in time $\mathcal{O}(|T_{\max}^t|)$ the t -th node $i_t = \arg \min_{i \in T_{\max}^t} \sigma(T_{\max}^t, i)$ selected by SEL. Let $N(i_t)$ be the set of all nodes adjacent to node i_t in T_{\max}^t . For all nodes $i' \in N(i_t)$, SEL compares $|T_{i'}|$ to the smallest key *bottom* stored in D . We have three cases:

1. If $|T_{i'}| \leq \text{bottom}$ and $\sigma(D) \geq Q - t$ then the algorithm does nothing, since $T_{i'}$ (or subtrees thereof) will never be largest in the subsequent steps of the construction of L_{SEL} , i.e., there will not exist any node $i_{t'}$ with $t' > t$ such that $i_{t'} \in T_{i'}$.
2. If $|T_{i'}| \leq \text{bottom}$ and $\sigma(D) < Q - t$, or if $|T_{i'}| > \text{bottom}$ and $\sigma(D) < Q$ then SEL inserts a pointer to i' together with the associated key $|T_{i'}|$. Note that, since D is not full (i.e., $\sigma(D) < Q$), the algorithm need not eliminate any item in D .
3. If $|T_{i'}| > \text{bottom}$ and $\sigma(D) = Q$ then SEL eliminates from D the item having the smallest key, and inserts a pointer to i' , together with the associated key $|T_{i'}|$.

Finally, SEL eliminates node i_t and all edges (both undirected and directed) incident to it. Note that this elimination implies that we can easily perform a depth-first visit within T_{\max}^s for each $s \leq Q$, since T_{\max}^s is always completely disconnected from the rest of the tree T .

In order to turn L_{SEL} into L_{SEL}^+ , the algorithm proceeds incrementally, using a technique borrowed from [7]. Just after the selection of the first node i_1 , a depth-first visit starting from i_1 is performed. During each backtracking step of this visit, the algorithm associates with each edge (i, j) , the closer node to i_1 between the two nodes i and j . In other words, SEL assigns a direction to each undirected edge (i, j) so as to be able to efficiently find the path connecting each given node i to i_1 . When the t -th node i_t is selected, SEL follows these edge directions from i_t towards i_1 . Let us denote by $\pi(i, j)$ the path connecting node i to node j . During the traversal of $\pi(i_1, i_t)$, the algorithm assigns a special mark to each visited node, until the algorithm reaches the first node $j \in \pi(i_1, i_t)$ which has already been marked. Let $\eta(i, L)$ be the maximum number of edge disjoint paths connecting i to nodes in the query set L . Observe that all nodes i for which $\eta(i, L_t) > \eta(i, L_{t-1})$ must necessarily belong to $\pi(i_t, j)$. We have $\eta(i_t, L_t) = 1$, and $\eta(i, L_t) = 2$, for all internal nodes i in the path $\pi(i_t, j)$. Hence, j is the unique node that we may need to add as a new fork node (if $j \notin \text{FORK}(L_{t-1})$). In fact, j is the unique node such that the number of edge-disjoint paths connecting it to query nodes may increase, and be actually larger than 2.

Therefore if $j \in L_{t-1}^+$ we need not add any fork node during the incremental construction of L_{SEL}^+ . On the other hand, if $j \notin L_{t-1}^+$ then $\eta(i, L_{t-1}) = 2$, which implies $\eta(i, L_t) = 3$. This is the case when SEL views j as new fork node to be added to the query set L_{SEL} under consideration.

In order to bound the total time required by SEL for selecting Q nodes, we rely on Lemma 4, showing that $|T_{\max}^t| \leq 2|V|/t$. The two depth-first visits performed for each node i_t take $\mathcal{O}(|T_{\max}^t|)$ steps. Hence the overall running time spent on the depth-first visits is $\mathcal{O}(\sum_{t \leq Q} 2|V|/t) = \mathcal{O}(|V| \log Q)$. The total time

⁶If $t = 1$ the priority deque D is empty.

⁷In the initial step $t = 1$ (i.e., when $T_{\max}^t \equiv T$) node r can be chosen arbitrarily.

spent for incrementally finding the fork nodes of L_{SEL} is linear in the number of nodes marked by the algorithm, which is equal to $|V|$. Finally, handling the priority deque D takes $|V|$ times the worst-case time for eliminating an item with the smallest (or largest) key or adding a new item. This is again $\mathcal{O}(|V| \log Q)$.

We now turn to the implementation of the prediction phase. PRED operates in two phases. In the first phase, the algorithm performs a depth-first visit of each hinge-tree \mathcal{T} , starting from each connection node (thereby visiting the nodes of all 1-hinge-tree once, and the nodes of all 2-hinge-tree twice). During these visits, we add to the nodes a tag containing (i) the label of node $i_{\mathcal{T}}$ from which the depth-first visit started, and (ii) the distance between $i_{\mathcal{T}}$ and the currently visited node. In the second phase, we perform a second depth-first visit, this time on the whole tree T . During this visit, we predict each node $i \in V \setminus L$ with the label coupled with smaller distance stored in the tags of⁸ i . The total time of these visits is linear in $|V|$ since each node of T gets visited at most 3 times.

7 Conclusions and ongoing work

The results proven in this paper characterize, up to constant factors, the optimal algorithms for adversarial active learning on trees in two main settings. In the first setting the goal is to minimize the number of mistakes on the non-queried vertices under a certain query budget. In the second setting the goal is to minimize the sum of queries and mistakes under no restriction on the number of queries.

An important open question is the extension of our results to the general case of active learning on graphs. While a direct characterization of optimality on general graphs is likely to require new analytical tools, an alternative line of attack is reducing the graph learning problem to the tree learning problem via the use of spanning trees. Certain types of spanning trees, such as random spanning trees, are known to summarize well the graph structure relevant to passive learning — see, e.g., [7]. In the case of active learning, however, we want good query sets on the graph to correspond to good query sets on the spanning tree, and random spanning trees may fail to do so in simple cases. For example, consider a set of m cliques connected through bridges, so that each clique is connected to, say, k other cliques. The breadth-first spanning tree of this graph is a set of connected stars. This tree clearly reveals a query set (the star centers) which is good for regular labelings (cfr., the binary system example of Section 1). On the other hand, for certain choices of m and k a random spanning tree has a good probability of hiding the clustered nature of the original graph, thus leading to the selection of bad query sets.

In order to gain intuition about this phenomenon, we are currently running experiments on various real-world graphs using different types of spanning trees, where we measure the number of mistakes made by our algorithm (for various choices of the budget size) against common baselines.

We also believe that an extension to general graphs of our algorithm does actually exist. However, the complexity of the methods employed in [6] suggests that techniques based on minimizing Ψ^* on general graphs are computationally very expensive.

Acknowledgments. This work was supported in part by Google Inc. through a Google Research Award and by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the authors' views.

References

- [1] P. Afshani, E. Chiniforooshan, R. Dorrigiv, A. Farzan, M. Mirzazadeh, N. Simjour, H. Zarrabi-Zadeh. On the complexity of finding an unknown cut via vertex queries. *COCOON 2007*, pages 459–469.
- [2] Belkin, M., Matveeva, I., and Niyogi, P. Regularization and semi-supervised learning on large graphs. *COLT 2004*, pages 624–638.
- [3] Bengio, Y., Delalleau, O., and Le Roux, N. Label propagation and quadratic criterion. In *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.
- [4] Blum, A. and Chawla, S. Learning from labeled and unlabeled data using graph mincuts. *ICML 2001*, pages 19–26.
- [5] A. Blum, J. Lafferty, R. Reddy, and M.R. Rwebangira. Semi-supervised learning using randomized mincuts. *ICML 2004*.
- [6] A. Guillory and J. Bilmes. Label Selection on Graphs. *NIPS 2009*.
- [7] N. Cesa-Bianchi, C. Gentile, F. Vitale. Fast and optimal prediction of a labeled tree. *COLT 2009*.
- [8] J. Katajainen, F. Vitale. Navigation piles with applications to sorting, priority queues, and priority deques. *Nordic Journal of Computing*, 10(3):238–262, 2003.
- [9] R. Lyons and Y. Peres. *Probability on Trees and Networks*. Manuscript, 2009.
- [10] Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.

⁸If i belongs to a 1-hinge-tree, we simply predict y_i with the unique label stored in the tag.

Adaptive Submodularity: A New Approach to Active Learning and Stochastic Optimization

Daniel Golovin
California Institute of Technology
Pasadena, CA 91125
dgolovin@caltech.edu

Andreas Krause
California Institute of Technology
Pasadena, CA 91125
krausea@caltech.edu

Abstract

Solving stochastic optimization problems under partial observability, where one needs to adaptively make decisions with uncertain outcomes, is a fundamental but notoriously difficult challenge. In this paper, we introduce the concept of *adaptive submodularity*, generalizing submodular set functions to adaptive policies. We prove that if a problem satisfies this property, a simple adaptive greedy algorithm is guaranteed to be competitive with the optimal policy. We illustrate the usefulness of the concept by giving several examples of adaptive submodular objectives arising in diverse applications including sensor placement, viral marketing and pool-based active learning. Proving adaptive submodularity for these problems allows us to recover existing results in these applications as special cases and leads to natural generalizations.

1 Introduction

In many natural optimization problems one needs to adaptively make a sequence of decisions, taking into account observations about the outcome of past decisions. Often, these outcomes are uncertain, and one may only know a probability distribution over them. Finding optimal policies for decision making in such partially observable stochastic optimization problems is notoriously intractable. In this paper, we analyze a particular class of partially observable stochastic optimization problems. We introduce the concept of *adaptive submodularity*, and prove that if a problem satisfies this property, a simple adaptive greedy algorithm is guaranteed to obtain near-optimal solutions. *Adaptive submodularity* generalizes the notion of submodularity¹, which has been successfully used to develop approximation algorithms for a variety of non-adaptive optimization problems. Submodularity, informally, is an intuitive notion of diminishing returns, which states that adding an element to a small set helps more than adding that same element to a larger (super-)set. A celebrated result of Nemhauser et al. (1978) guarantees that for such submodular functions, a simple greedy algorithm, which adds the element that maximally increases the objective value, selects a near optimal set of k elements. The challenge in generalizing submodularity to adaptive planning is that feasible solutions are now policies (decision trees) instead of subsets. We consider a natural analog of the diminishing returns property for adaptive problems, which reduces to the classical notion of submodular set functions for deterministic distributions. We show how the results of Nemhauser et al. generalize to the adaptive setting. We further demonstrate the usefulness of the concept by showing how it captures known results in stochastic optimization and active learning as special cases, and leads to natural generalizations.

As a first example, consider the problem of deploying a collection of sensors to monitor some spatial phenomenon. Each sensor can cover a region depending on its sensing range. Suppose we would like to find the best subset of k locations to place the sensors. In this application, intuitively, adding a sensor helps more if we have placed few sensors so far and helps less if we have already placed many sensors. We can formalize this diminishing returns property using the notion of submodularity – the total area covered by the sensors is a submodular function defined over all sets of locations. Krause and Guestrin (2007) show that many more realistic utility functions in sensor placement (such as the improvement in prediction accuracy w.r.t. some probabilistic model) are submodular as well. Now consider the following stochastic variant: Instead of deploying a fixed set of sensors, we deploy one sensor at a time. With a certain probability, deployed sensors can fail, and our goal is to maximize the area covered by the functioning sensors. Thus, when deploying the next sensor, we need to take into account which of the sensors we deployed in the past failed. This problem

¹For an extensive treatment of submodularity, see the books of Fujishige (1991) and Schrijver (2003).

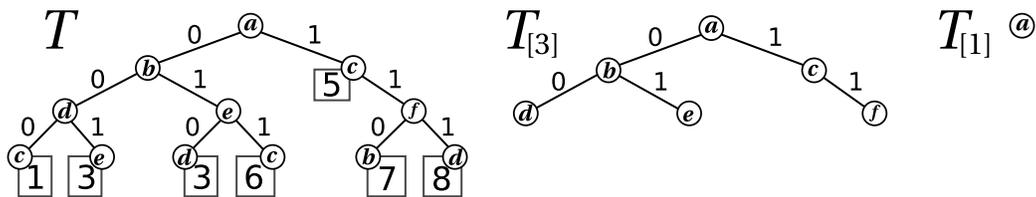


Figure 1: Left: Example policy tree, with edges labelled by state and rewards at (potential) terminal nodes in the rectangles. Middle and right: Prunings of policy trees T at layers 3 and 1.

has been studied by Asadpour et al. (2008) for the case where each sensor fails independently at random. In this paper, we show that the coverage objective is adaptive submodular, and use this concept to handle more general settings (where, e.g., rather than all-or-nothing failures there are different types of sensor failures of varying severity).

As another example, consider a viral marketing problem, where we are given a social network, and we want to influence as many people as possible in the network to buy some product. We do that by giving the product for free to a subset of the people, and hope that they convince their friends to buy the product as well. Formally, we have a graph, and each edge e is labeled by a number $0 \leq p_e \leq 1$. We “influence” a subset of nodes in the graph, and for each influenced node, their neighbors get randomly influenced according to the probability annotated on the edge connecting the nodes. This process repeats, until no further node gets influenced. Kempe et al. (2003) show that the set function which quantifies the expected number of nodes influenced is submodular. A natural stochastic variant of the problem is where we pick a node, get to see which nodes it influenced, then adaptively pick the next node based on these observations and so on. We show that a large class of such adaptive influence maximization problems satisfies adaptive submodularity.

Our third application is in pool-based active learning, where we are given an unlabeled data set, and we would like to adaptively pick a small set of examples whose labels imply all other labels. Thus, we want to pick examples to shrink the remaining version space (the set of consistent hypotheses) as quickly as possible. Here, we show that the reduction in version space mass is adaptive submodular, and use that observation to prove that the adaptive greedy algorithm is a near-optimal querying policy, recovering and generalizing results by Kosaraju et al. (1999) and Dasgupta (2004). Our results for active learning are also related to recent results of Guillory and Bilmes (2010) who study a generalization of submodular set cover to an interactive setting. In contrast to our approach however, Guillory and Bilmes (2010) analyze worst-case costs, and use rather different technical definitions and proof techniques.

In summary, our main contributions are:

- We consider a particular class of adaptive stochastic optimization problems, which we prove to be hard to approximate in general.
- We introduce the concept of *adaptive submodularity*, and prove that if a problem instance satisfies this property, a simple adaptive greedy policy performs near-optimally.
- We illustrate adaptive submodularity on several realistic problems, including Stochastic Maximum Coverage, Adaptive Viral Marketing and Active Learning. For these applications, adaptive submodularity allows us to recover known results and prove natural generalizations.

2 Adaptive Stochastic Optimization

Let E be a finite set of items. Each item $e \in E$ is in a particular state $\Phi(e) \in O$ from a set O of possible states. Hereby, $\Phi : E \rightarrow O$ is a (random) *realization* of the ground set, indicating which state each item is in. We take a Bayesian approach and assume that there is a (known) probability distribution $\mathbb{P}[\Phi]$ over realizations. We will consider the problem where we sequentially pick an item $e \in E$, get to see its state $\Phi(e)$, pick the next item, get to see its state, and so on. After each pick, our observations so far can be represented as a *partial realization* $\Psi \subseteq E \times O$, a function from some subset of E (i.e., the set of items that we already picked) to their states. A partial realization Ψ is *consistent* with a realization Φ if they are equal everywhere in the domain of Ψ . In this case we write $\Phi \sim \Psi$. If Ψ and Ψ' are both consistent with some Φ , and $\text{dom}(\Psi) \subset \text{dom}(\Psi')$, we say Ψ is a *subrealization* of Ψ' .

We encode our adaptive strategy for picking items as a *policy* π , which is a function from a set of partial realizations to E , specifying which item to pick next under a particular set of observations. If $\Psi \notin \text{dom}(\pi)$, the policy terminates (stops picking items) upon observation of Ψ . Technically, we require that the domain of π is closed under subrealizations. That is, if $\Psi' \in \text{dom}(\pi)$ and Ψ is a subrealization of Ψ' then $\Psi \in \text{dom}(\pi)$. This condition simply ensures that the decision tree T^π associated with π as described below is connected. We define both $E(\pi, \Phi)$ and $E(T^\pi, \Phi)$ as the set of items picked by π conditioned on realization Φ . We also

allow randomized policies that are functions from a set of partial realizations to distributions on E .

Each deterministic policy π can be associated with a tree T^π in a natural way (see Fig. 1 (left) for an illustration). We create the root of T^π , and label it with a tuple consisting of a partial realization \emptyset and an item $\pi(\emptyset)$. Then inductively for each node, if its label is (Ψ, e) , we construct a child for it for each state x such that $\Psi \cup \{(e, x)\} \in \text{dom}(\pi)$, labeled with $(\Psi \cup \{(e, x)\}, \pi(\Psi \cup \{(e, x)\}))$. A missing child for state x simply means that the policy terminates (stops picking items upon observing x). Thus, the first coordinate of the label at a node indicates what is known when the policy reaches that node, and the second coordinate indicates what it will do next. Similarly, randomized policies can be associated with distributions over trees in a natural way.

We wish to maximize, subject to some constraints, a utility function $f : 2^E \times O^E \rightarrow \mathbb{R}_{\geq 0}$ that depends on which items we pick and which state each item is in. Based on this notation, the expected utility of a policy π is $f_{\text{avg}}(\pi) := \mathbb{E}_\Phi[f(E(\pi, \Phi), \Phi)]$. The goal of the *Adaptive Stochastic Maximization* problem is to find a policy π^* such that

$$\pi^* \in \arg \max_{\pi} f_{\text{avg}}(\pi) \text{ subject to } |E(\pi, \Phi)| \leq k \text{ for all } \Phi, \quad (1)$$

where k is a budget on how many items can be picked.

Unfortunately, as we will show in §8, even for linear functions f , i.e., those where $f(A, \Phi) = \sum_{e \in A} w_{e, \Phi}$ is simply the sum of weights (depending on the realization Φ), Problems (1) is hard to approximate under reasonable complexity theoretic assumptions. Despite the hardness of the general problem, in the following sections we will identify conditions that are sufficient to allow us to approximately solve it.

Incorporating Item Costs. Instead of quantifying the cost of a set $E(\pi, \Phi)$ by the number of elements $|E(\pi, \Phi)|$, we can also consider the case where each item $e \in E$ has a cost $c(e)$. We show how to handle this extension and give many other results in the extended version of this paper (Golovin & Krause, 2010).

3 Adaptive Submodularity

We first review the classical notion of submodular set functions, and then introduce the novel notion of adaptive submodularity.

Submodularity. Let us first consider the simple special case where $\mathbb{P}[\Phi]$ is deterministic or, equivalently, $|O| = 1$. In this case, the realization Φ is known to the decision maker in advance, and thus there is no benefit in adaptive selection. Thus, Problem (1) is equivalent to finding a set $A^* \subseteq E$ such that

$$A^* \in \arg \max_{A \subseteq E} f(A, \Phi) \text{ such that } |A| \leq k. \quad (2)$$

For most interesting classes of utility functions f , this is an NP-hard optimization problem. However, in many practical problems, such as those mentioned in §1, $f(A) = f(A, \Phi)$ satisfies *submodularity*. A set function $f : 2^E \rightarrow \mathbb{R}$ is called submodular if, whenever $A \subseteq B \subseteq E$ and $e \in E \setminus B$ it holds that

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B), \quad (3)$$

i.e., adding e to the smaller set A increases f at least as much as adding e to the superset B . Furthermore, f is called *monotone*, if, whenever $A \subseteq B$ it holds that $f(A) \leq f(B)$. A celebrated result by Nemhauser et al. (1978) states that for monotone submodular functions with $f(\emptyset) = 0$, a simple greedy algorithm that starts with the empty set, $A_0 = \emptyset$ and chooses $A_{i+1} = A_i \cup \{\arg \max_{e \in E \setminus A_i} f(A_i \cup \{e\})\}$ guarantees that $f(A_k) \geq (1 - 1/e) \max_{|A| \leq k} f(A)$. Thus, the greedy set A_k obtains at least a $(1 - 1/e)$ fraction of the optimal value achievable using k elements. Furthermore, Feige (1998) shows that this result is tight if $P \neq NP$; under this assumption no polynomial time algorithm can achieve a $(1 - 1/e + \epsilon)$ -approximation for any constant $\epsilon > 0$, even for the special case of Maximum k -Cover where $f(A)$ is the cardinality of the union of sets indexed by A .

Now let us relax the assumption that $\mathbb{P}[\Phi]$ is deterministic. In this case, we may still want to find a non-adaptive solution (i.e., a constant policy π_A that always picks set A independently of Φ) maximizing $f_{\text{avg}}(\pi_A)$. If f is *pointwise* submodular, i.e., $f(A, \Phi)$ is submodular in A for any fixed Φ , the function $f(A) = f_{\text{avg}}(\pi_A)$ is submodular, since nonnegative linear combinations of submodular functions remain submodular. Thus, the greedy algorithm allows us to find a near-optimal *non-adaptive* policy.

However, in practice, we may be more interested in obtaining a non-constant policy π , that *adaptively* chooses items based on previous observations. Thus, the question is whether there is a natural extension of submodularity to policies. In the following, we will develop such a notion – *adaptive submodularity*.

Adaptive submodularity. The key challenge is to find an appropriate generalization of the diminishing returns condition (3). Informally, our generalization will require that playing a layer k of a policy tree T^π earlier in the policy cannot decrease its marginal contribution to the objective. Since there are many more nodes at layer k than at earlier layers, we consider playing an appropriate distribution at earlier layers to make the comparison formal.

We will now formalize the above intuition. Given a tree $T = T^\pi$ we define its *level- k -pruning* $T_{[k]}$ as the subtree of T induced on all nodes of depth k or less, i.e., those that can be reached from the root via a path of at most $k - 1$ edges. Tree pruning is illustrated in Fig. 1. Given two policies π_1, π_2 associated with trees T_1 and T_2 we define $\pi_1 @ \pi_2$ as the policy obtained by running π_1 to completion, and then running policy π_2 as if from a fresh start, ignoring the information gathered during the running of π_1 . We let $T_1 @ T_2$ denote the tree associated with policy $\pi_1 @ \pi_2$. This concept is illustrated in Fig. 2.

Fix any integers i and j so that $0 \leq i < j$, and any policy π . Define $\mathcal{D}(T, \Psi, j)$ to be the distribution on E induced by executing T under a random realization which is consistent with Ψ , and then outputting the item selected at depth j in T . For any node u of $T = T^\pi$, let $e(u)$ denote the item selected by u , and let Ψ_u be the partial realization encoding all state observations known to T just as it reaches u . As illustrated in Fig. 3, let $T_{[i] \cup \{j\}}^\pi$ be the (random) tree obtained as follows: Start with $T_{[i]}$ and for each of its leaves u and every possible state o (i.e., those with $\mathbb{P}[\Phi(e(u)) = o \mid \Phi \sim \Psi_u] > 0$) connect u to a new node which plays a (random) item e drawn from $\mathcal{D}(T, \Psi_u \cup \{(e(u), o)\}, j)$. The new node's corresponding partial realization, indicating what is known when it is first reached, is $\Psi_u \cup \{(e(u), o)\}$. Note that if T terminates before selecting j items for some realizations consistent with Ψ , then $\mathcal{D}(T, \Psi, j)$ will select nothing at all with the total conditional probability mass of such realizations.

We now introduce our generalizations of monotonicity and submodularity to the adaptive setting:

Definition 1 (Adaptive Monotonicity) A function $f : 2^E \times O^E \rightarrow \mathbb{R}_{\geq 0}$ is adaptive monotone with respect to distribution $\mathbb{P}[\Phi]$ if for all policies π, π' it holds that $f_{\text{avg}}(\pi) \leq f_{\text{avg}}(\pi' @ \pi)$, where $f_{\text{avg}}(\pi) := \mathbb{E}_\Phi[f(E(\pi, \Phi), \Phi)]$ is defined w.r.t. $\mathbb{P}[\Phi]$.

Definition 2 (Adaptive Submodularity) A function $f : 2^E \times O^E \rightarrow \mathbb{R}_{\geq 0}$ is adaptive submodular with respect to distribution $\mathbb{P}[\Phi]$ if for all policies π and for all $0 \leq i < j$

$$f_{\text{avg}}(T_{[j]}^\pi) - f_{\text{avg}}(T_{[j-1]}^\pi) \leq \mathbb{E} \left[f_{\text{avg}}(T_{[i] \cup \{j\}}^\pi) - f_{\text{avg}}(T_{[i]}^\pi) \right] \quad (4)$$

where the expectation is over the random choice of $T_{[i] \cup \{j\}}^\pi$ and f_{avg} is defined w.r.t. $\mathbb{P}[\Phi]$.

We will give concrete examples of adaptive monotone and adaptive submodular functions that arise in the applications introduced in §1 in §5, §6 and §7. It turns out there is an equivalent characterization of adaptive submodular functions in terms of derivatives of the expected value with respect to each item e , conditioned on the states of the previously selected items. We denote this derivative by $\Delta_\Psi(e)$, where Ψ is the current partial realization. Formally,

$$\Delta_\Psi(e) := \mathbb{E}_\Phi[f(\text{dom}(\Psi) \cup \{e\}, \Phi) - f(\text{dom}(\Psi), \Phi) \mid \Phi \sim \Psi]. \quad (5)$$

Proposition 3 A function $f : 2^E \times O^E \rightarrow \mathbb{R}_{\geq 0}$ is adaptive submodular if and only if for all Ψ and Ψ' such that Ψ is a subrealization of Ψ' (i.e., $\Psi \subseteq \Psi'$), and for all e , we have $\Delta_{\Psi'}(e) \leq \Delta_\Psi(e)$.

Proof: (\Rightarrow) To get from Eq. (4) to $\Delta_{\Psi'}(e) \leq \Delta_\Psi(e)$, generate an order \prec for $\text{dom}(\Psi')$ such that each item in $\text{dom}(\Psi)$ is less than each item in $\text{dom}(\Psi') \setminus \text{dom}(\Psi)$. Let e_1, e_2, \dots, e_m be the items of $\text{dom}(\Psi')$ in order of \prec . Let $e_{m+1} := e$. Define a policy tree T that is a path $u_1, u_2, \dots, u_m, u_{m+1}$ where each u_i is labeled with partial realization $\{(e_j, \Psi'(e_j)) : j < i\}$ and selects item e_i . Then applying Eq. (4) with T and $i = |\text{dom}(\Psi)|$, $j = m + 1$ yields $\mathbb{P}[\Psi' \mid \Psi] \cdot \Delta_{\Psi'}(e) \leq \mathbb{P}[\Psi' \mid \Psi] \cdot \Delta_\Psi(e)$ and hence $\Delta_{\Psi'}(e) \leq \Delta_\Psi(e)$.

(\Leftarrow) Informally, if $\Delta_{\Psi'}(e) \leq \Delta_\Psi(e)$ for all $\Psi \subseteq \Psi'$ and e , then in any tree T moving items from layer j up to layer i cannot decrease their marginal benefit. Since each item e in layer j of T is selected with the same probability in $T_{[j]}^\pi$ and in $T_{[i] \cup \{j\}}^\pi$, this implies Eq. (4). ■

Properties of adaptive submodular functions. It can be seen that adaptive monotonicity and adaptive submodularity enjoy similar closure properties as monotone submodular functions. In particular, if $w_1, \dots, w_m \geq 0$ and f_1, \dots, f_m are adaptive monotone submodular w.r.t. distribution $\mathbb{P}[\Phi]$, then $f(A, \Phi) = \sum_{i=1}^m w_i f_i(A, \Phi)$ is adaptive monotone submodular w.r.t. $\mathbb{P}[\Phi]$. Similarly, for a fixed constant $c \geq 0$ and adaptive monotone submodular function f , the function $g(E, \Phi) = \min(f(E, \Phi), c)$ is adaptive monotone submodular. Thus, adaptive monotone submodularity is preserved by nonnegative linear combinations and by truncation.

4 Guarantee for the Greedy Policy

The greedy policy π_{greedy} at each time step tries to myopically increase the expected objective value, given its current observations. That is, suppose $f : 2^E \times O^E \rightarrow \mathbb{R}_{\geq 0}$ is the objective, and Ψ is the partial realization indicating the states of items selected so far. Then the greedy policy will select the item e maximizing the

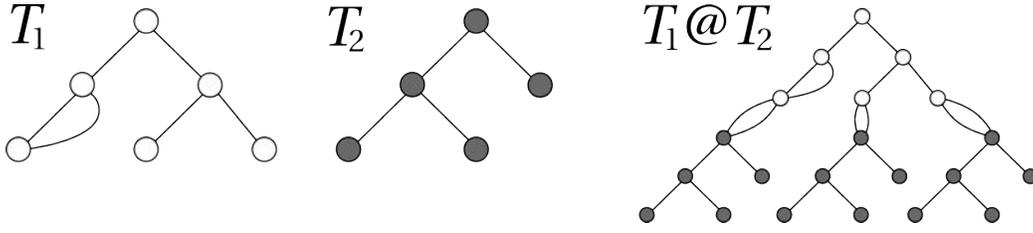


Figure 2: Concatenation of policy trees.

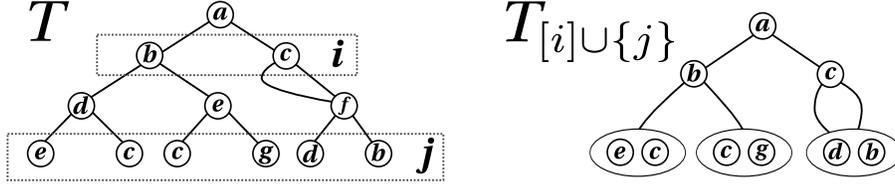


Figure 3: The “collapsed” tree $T_{[i] \cup \{j\}}$ which, speaking informally, plays layer j of T in its layer $i + 1$. Each leaf of $T_{[i] \cup \{j\}}$ is depicted as containing the set of items it samples from. The diminishing returns condition for adaptive submodularity states that the marginal benefit of the j^{th} layer in T may not exceed the marginal benefit of the $(i + 1)^{\text{th}}$ layer in $T_{[i] \cup \{j\}}$. The latter quantity is the marginal benefit layer $(i + 1)$ would obtain if each layer $(i + 1)$ node u selected its item from the distribution on items selected at layer j by executions of T that reach u . For example, if T has a 30% chance of picking e as its last item, conditioned on it reaching the layer $(i + 1)$ node labeled d , then the left-most leaf of $T_{[i] \cup \{j\}}$ picks e with 30% probability and picks c with 70% probability. Alternately, if we use the convention that left edges correspond to the selected item being in state 0, and right edges correspond to the selected item being in state 1 (as depicted in Fig. 1), and let $\Psi = \{(a, 0), (b, 0)\}$, then we may say that the left-most leaf of $T_{[i] \cup \{j\}}$ picks an item from distribution $\mathcal{D}(T, \Psi, j)$, so that it picks e with probability $\mathbb{P}_{\Phi} [T \text{ picks } e \text{ in layer } j \mid \Phi \sim \Psi] = 0.3$, and picks c with probability $\mathbb{P}_{\Phi} [T \text{ picks } c \text{ in layer } j \mid \Phi \sim \Psi] = 0.7$.

expected increase in value, conditioned on the observed states of items it has already selected (i.e., conditioned on $\Phi \sim \Psi$). That is, it will select e to maximize the quantity $\Delta_{\Psi}(e)$ defined in Eq. (5).

In some applications, finding an item maximizing $\Delta_{\Psi}(e)$ may be computationally intractable, and the best we can do is find an α -approximation to the best greedy move. This means we find an e' such that $\Delta_{\Psi}(e') \geq \frac{1}{\alpha} \max_e \Delta_{\Psi}(e)$. We call a policy which always selects such an item an α -approximate greedy policy.

In this section we establish that if the objective function is adaptive submodular with respect to the distribution describing the environment in which we operate, then the greedy policy and any α -approximate greedy policy inherit precisely the performance guarantees of the greedy and α -approximate greedy algorithms for classic (nonadaptive) submodular maximization. We have the following result.

Theorem 4 Fix any $\alpha \geq 1$. If f is adaptive monotone and adaptive submodular with respect to the distribution $\mathbb{P}[\Phi]$, and π is an α -approximate greedy policy, then for all policies π^* and positive integers ℓ, k

$$f_{\text{avg}}(T_{[\ell]}^{\pi}) > \left(1 - e^{-\ell/\alpha k}\right) f_{\text{avg}}(T_{[k]}^{\pi^*}).$$

In particular, with $\ell = k$ this implies any α -approximate greedy policy achieves a $(1 - e^{-1/\alpha})$ approximation to the expected reward of the best policy, if both are terminated after running for an equal number of steps.

Proof: The proof goes along the lines of the performance analysis of the greedy algorithm for maximizing a submodular function subject to a cardinality constraint found in Nemhauser et al. (1978). An extension of that analysis to α -approximate greedy algorithms, which is analogous to ours but for the nonadaptive case, is shown by Goundan and Schulz (2007). Let $T = T_{[\ell]}^{\pi}, T^* = T_{[k]}^{\pi^*}$. Then for all $i, 0 \leq i < \ell$

$$f_{\text{avg}}(T^*) \leq f_{\text{avg}}(T_{[i]} \circledast T^*) \tag{6}$$

$$= f_{\text{avg}}(T_{[i]}) + \sum_{j=1}^k \left(f_{\text{avg}}(T_{[i]} \circledast T_{[j]}^*) - f_{\text{avg}}(T_{[i]} \circledast T_{[j-1]}^*) \right) \tag{7}$$

$$\leq f_{\text{avg}}(T_{[i]}) + \sum_{j=1}^k \mathbb{E} \left[f_{\text{avg}} \left((T_{[i]} \circledast T^*)_{[i] \cup \{i+j\}} \right) - f_{\text{avg}}(T_{[i]}) \right] \tag{8}$$

$$\leq f_{\text{avg}}(T_{[i]}) + \alpha \sum_{j=1}^k \left(f_{\text{avg}}(T_{[i+1]}) - f_{\text{avg}}(T_{[i]}) \right) \tag{9}$$

The first inequality is due to the adaptive monotonicity of f , from which we may infer $f_{\text{avg}}(T_2) \leq f_{\text{avg}}(T_1 @ T_2)$ for any T_1 and T_2 . The second is a simple telescoping sum. The third is a direct application of the adaptive submodularity guarantee of f with $T_{[i]} @ T_{[j]}^*$ at levels i and $i + j$, and the fourth is by the definition of an α -approximate greedy policy. Now define $\Delta_i := f_{\text{avg}}(T^*) - f_{\text{avg}}(T_{[i]})$, so that Eq. (9) implies $\Delta_i \leq \alpha k (\Delta_i - \Delta_{i+1})$, from which we infer $\Delta_{i+1} \leq (1 - \frac{1}{\alpha k}) \Delta_i$ and hence $\Delta_\ell \leq (1 - \frac{1}{\alpha k})^\ell \Delta_0 < e^{-\ell/\alpha k} \Delta_0$, where for this last inequality we have used the fact that $1 - x < e^{-x}$ for all $x > 0$. Thus $f_{\text{avg}}(T^*) - f_{\text{avg}}(T_{[\ell]}) < e^{-\ell/\alpha k} (f_{\text{avg}}(T^*) - f_{\text{avg}}(T_{[0]})) \leq e^{-\ell/\alpha k} f_{\text{avg}}(T^*)$ so $f_{\text{avg}}(T) > (1 - e^{-\ell/\alpha k}) f_{\text{avg}}(T^*)$. ■

Note that if the greedy rule can be implemented only with small *absolute* error rather than small *relative* error, i.e., $\Delta_\Psi(e') \geq \max_e \Delta_\Psi(e) - \varepsilon$, a similar argument shows that

$$f_{\text{avg}}(T_{[\ell]}^\pi) \geq \left(1 - e^{-\ell/k}\right) f_{\text{avg}}(T_{[k]}^{\pi^*}) - \ell\varepsilon.$$

This is important, since small absolute error can always be achieved (with high probability) whenever f can be evaluated efficiently, and sampling $P(\Phi | \Psi)$ is efficient. In this case, we can approximate

$$\Delta_\Psi(e) \approx \frac{1}{N} \sum_{i=1}^N [f(\text{dom}(\Psi) \cup \{e\}, \Phi_i) - f(\text{dom}(\Psi), \Phi_i)],$$

where Φ_i are sampled i.i.d. from $P(\Phi | \Psi)$. Note that the characterization of adaptive submodularity in Proposition 3 allows us to implement an “accelerated” version of the adaptive greedy algorithm using lazy evaluations of marginal benefits as originally suggested for the nonadaptive case by Minoux (1978).

5 Application: Stochastic Submodular Maximization

As our first application, consider the sensor placement problem introduced in §1. Suppose we would like to monitor a spatial phenomenon such as temperature in a building. We discretize the environment into a set E of locations. We would like to pick a subset $A \subseteq E$ of k locations that is most “informative”, where we use a set function $\hat{f}(A)$ quantifying the informativeness of placement A . Krause and Guestrin (2007) show that many natural objective functions (such as reduction in predictive uncertainty measured in terms of Shannon entropy) are monotone submodular.

Now consider the problem, where sensors can fail or partially fail (e.g., be subject to some varying amount of noise) after deployment. We can model this extension by assigning a state $\Phi(e) \in O$ to each possible location, indicating the extent to which a sensor placed at location e is working. To quantify the value of a set of sensor deployments under a realization Φ indicating to what extent the various sensors are working, we first define (e, o) for each $e \in E$ and $o \in O$, which represents the placement of a sensor in state o at location e . We then suppose there is a function $\hat{f} : 2^{E \times O} \rightarrow \mathbb{R}_{\geq 0}$ which quantifies the informativeness of a set of sensor deployments in arbitrary states. The utility $f(A, \Phi)$ of placing sensors at the locations in A under realization Φ then is

$$f(A, \Phi) = \hat{f}(\{(e, \Phi(e)) : e \in A\}).$$

We aim to adaptively place k sensors to maximize our expected utility. We assume that sensor failures at each location are independent of each other, i.e., $\mathbb{P}[\Phi] = \prod_e \mathbb{P}[\Phi(e)]$, where $\mathbb{P}[\Phi(e) = o]$ is the probability that a sensor placed at location e will be in state o . Goemans and Vondrák (2006) studied a related problem called *Stochastic Covering* where the goal is to achieve the maximum attainable objective value at minimum cost, i.e., their problem generalizes Set Cover in the same way our problem generalizes Maximum k -Cover. Asadpour et al. (2008) studied a special case of our problem, in which sensors either fail completely (in which case they contribute no value at all) or work perfectly, under the name *Stochastic Submodular Maximization*. They proved that the adaptive greedy algorithm obtains a constant fraction $(1 - 1/e)$ approximation to the optimal adaptive policy, provided \hat{f} is monotone submodular. We extend their result to multiple types of failures by showing that $f(A, \Phi)$ is adaptive submodular with respect to distribution $\mathbb{P}[\Phi]$ and then invoking Theorem 4.

Theorem 5 Fix a prior such that $\mathbb{P}[\Phi] = \prod_{e \in E} \mathbb{P}[\Phi(e)]$, and integer k and let the objective function $\hat{f} : 2^{E \times O} \rightarrow \mathbb{R}_{\geq 0}$ be monotone submodular. Let π be the adaptive greedy policy attempting to maximize f , and let π^* be any policy. Then

$$f_{\text{avg}}(T_{[k]}^\pi) \geq \left(1 - \frac{1}{e}\right) f_{\text{avg}}(T_{[k]}^{\pi^*}).$$

Proof: We prove Theorem 5 by first proving f is adaptive monotone and adaptive submodular in this model, and then applying Theorem 4. Adaptive monotonicity is readily proven after observing that $f(\cdot, \Phi)$ is monotone

for each Φ , and noting that for any Φ, T_1 and T_2 , we have $E(T_2, \Phi) \subseteq E(T_1 @ T_2, \Phi)$. Moving on to adaptive submodularity, fix any T and $i < j$. We prove Eq. (4) using the alternate characterization of Proposition 3. We use a coupled distribution over the realizations seen when running $T_{[j]}$ and $T_{[i] \cup \{j\}}$, such that the same realization is sampled for both. For any partial realization Ψ encoding the observations made immediately before reaching a level $i + 1$ node, and any ground set item e such that e is in the support of $\mathcal{D}(T, \Psi, j)$, consider the expected marginal contribution of e to the objective conditioned on the fact that the policy has observed Ψ for trees T and $T_{[i] \cup \{j\}}$. In both cases e is equally likely to be selected by the policy, and is equally likely to be in any given state, since $\Phi(e)$ is independent of $\{\Phi(e') : e' \in E \setminus \{e\}\}$. However, its marginal contribution under $T_{[j]}$ can be at most that under $T_{[i] \cup \{j\}}$ by the submodularity of \hat{f} , since in the former case there are potentially more items in the base set to which we add $(e, \Phi(e))$ (namely, the realized versions $(e', \Phi(e'))$ of those items e' selected in layers $i + 1$ through $j - 1$), but there are never fewer items in it. ■

6 Application: Adaptive Viral Marketing

For our next application, consider the following scenario. Suppose we would like to generate demand for a genuinely novel product. Potential customers do not realize how valuable the new product will be in their lives, and conventional advertisements are failing to induce them to try it. In this case, we may try to spur demand by offering a special promotional deal to a select few people, and hope that demand builds virally, propagating through the social network as people recommend the product to their friends and associates. Supposing we know something about the structure of the social networks people inhabit, and how ideas, innovation, and new product adoption diffuse through them, this begs the question: to which initial set of people should we offer the promotional deal, in order to spur maximum demand for our product? We imagine there is a fixed budget for the promotional campaign, which can be interpreted as a budget k indicating the maximum size of the initial set of people.

This, broadly, is the viral marketing problem. In the adaptive variant, we may select a person to offer the promotion to, make some observations about the resulting spread of demand for our product, and repeat. The same problem arises in the context of spreading technological, cultural, and intellectual innovations, broadly construed. In the interests of having unified terminology we follow Kempe et al. (2003) and talk of spreading *influence* through the social network, where we say people are *active* if they have adopted the idea or innovation in question, and *inactive* otherwise, and that a *influences* b if a convinces b to adopt the idea or innovation in question.

There are many ways to model the diffusion dynamics governing the spread of influence in a social network. We consider a basic and well-studied model, the *independent cascade model*, described in detail below. For this model Kempe et al. (2003) obtained a very interesting result; they showed that the eventual spread of the influence f (i.e., the ultimate number of customers that demand the product) is a monotone submodular function of the seed set S of initial people. This, in conjunction with the results of Nemhauser et al. (1978) implies that the natural greedy algorithm obtains at least $(1 - \frac{1}{e})$ of the value of the best feasible seed set, $\arg \max_{S: |S| \leq k} f(S)$. In this section, we use the idea of adaptive submodularity to extend their results in two directions simultaneously. First, we extend the guarantees to the adaptive version of the problem, and show that the greedy policy obtains at least $(1 - \frac{1}{e})$ of the value of the best *policy*. Second, we achieve this guarantee not only for the case where our reward is simply the number of influenced people, but also for any (nonnegative) monotone submodular function of the *set* of people influenced.

Independent Cascade Model. In this model, the social network is a directed graph $G = (V, A)$ where each vertex in V is a person, and each edge $(u, v) \in A$ has an associated binary random variable X_{uv} indicating if u will influence v . That is, $X_{uv} = 1$ if u will influence v once it has been influenced, and $X_{uv} = 0$ otherwise. The random variables X_{uv} are independent, and have known means $p_{uv} := \mathbb{E}[X_{uv}]$. We will call an edge (u, v) with $X_{uv} = 1$ a *live edge* and an edge with $X_{uv} = 0$ a *dead edge*. When a node u is activated, the edges X_{uv} to each neighbor v of u are sampled, and v is activated if (u, v) is live. Influence can then spread from u 's neighbors to their neighbors, and so on, according to the same process. Once active, nodes remain active throughout the process, however Kempe et al. (2003) show that this assumption is without loss of generality, and can be removed.

The Feedback Model. In the Adaptive Viral Marketing problem under the independent cascades model, the items correspond to people we can “activate” by, e.g., offering them the promotional deal. How we define the states $\Phi(u)$ depends on what information we obtain as a result of activating u . Given the nature of the diffusion process, activating u can have wide-ranging effects, so the state $\Phi(u)$ has more to do with the state of the social network on the whole than with u in particular. Specifically, we model $\Phi(u)$ as a function $\Phi_u : A \rightarrow \{0, 1, ?\}$, where $\Phi_u((u, v)) = 0$ means that activating u has revealed that (u, v) is dead, $\Phi_u((u, v)) = 1$ means that activating u has revealed that (u, v) is live, and $\Phi_u((u, v)) = ?$ means that activating u has not revealed the status of (u, v) (i.e., the value of X_{uv}). We require each realization to be *consistent* and *complete*. Consistency

means that no edge should be declared both live and dead by any two states. That is, for all $u, v \in V$ and $a \in A$, $(\Phi_u(a), \Phi_v(a)) \notin \{(0, 1), (1, 0)\}$. Completeness means that the status of each edge is revealed by some activation. That is, for all $a \in A$ there exists $u \in V$ such that $\Phi_u(a) \in \{0, 1\}$. A consistent and complete realization thus encodes X_{uv} for each edge (u, v) . Let $A(\Phi)$ denote the live edges as encoded by Φ . There are several candidates for which edge sets we are allowed to observe when activating a node u . We consider the following two concrete feedback models:

Myopic Feedback: After activating u we get to see the status (live or dead) of all edges exiting u in the social network, i.e., $\partial_+(u) := \{(u, v) : v \in V\} \cap A$.

Full-Adoption Feedback: After activating u we get to see the status (live or dead) of all edges exiting v , for all nodes v reachable from u via live edges (i.e., reachable from u in $(V, A(\Phi))$), where Φ is the true realization.

The Objective Function. In the simplest case, the reward for influencing a set $U \subseteq V$ of nodes is $\hat{f}(U) := |U|$. Kempe et al. (2003) obtain an $(1 - \frac{1}{e})$ -approximation for the slightly more general case in which each node u has a weight w_u indicating its importance, and the reward is $\hat{f}(U) := \sum_{u \in U} w_u$. We generalize this result further, to include arbitrary nonnegative monotone submodular reward functions \hat{f} . This allows us, for example, to encode a value associated with the *diversity* of the set of nodes influenced, such as the notion that it is better to achieve 20% market penetration in five different (equally important) demographic segments than 100% market penetration in one and 0% in the others.

Comparison with Stochastic Submodular Maximization. It is worth contrasting the Adaptive Viral Marketing problem with the Stochastic Submodular Maximization problem of §5. In the latter problem, we can think of the items as being random independently distributed sets. In Adaptive Viral Marketing by contrast, the random sets (of nodes influenced when a fixed node is selected) depend on the random status of the edges, and hence may be correlated through them. Nevertheless, we can obtain the same $(1 - \frac{1}{e})$ approximation factor for both problems.

We are now ready to formally state our result for this section.

Theorem 6 *The greedy policy obtains at least $(1 - \frac{1}{e})$ of the value of the best policy for the Adaptive Viral Marketing problem with arbitrary monotone submodular reward functions, in the independent cascade model, in both feedback models discussed above. That is, if $\sigma(S, \Phi)$ is the set of all activated nodes when S is the seed set of activated nodes and Φ is the realization, $\hat{f} : 2^V \rightarrow \mathbb{R}_{\geq 0}$ is an arbitrary monotone submodular function indicating the reward for influencing a set, and the objective function is $f(S, \Phi) := \hat{f}(\sigma(S, \Phi))$, then*

$$f_{\text{avg}}(T_{[k]}^{\text{greedy}}) \geq \left(1 - \frac{1}{e}\right) f_{\text{avg}}(T_{[k]})$$

for all $k \in \mathbb{N}$, where T^{greedy} is the policy tree of the greedy policy, and T is any policy tree.

Proof: It suffices to prove that f is adaptive submodular with respect to the probability distribution on realizations $\mathbb{P}[\Phi]$, in both feedback models, because then we can invoke Theorem 4 to complete the proof.

We will say we have *observed* an edge (u, v) if we know its status, i.e., if it is live or dead. We will actually prove that f is adaptive submodular in any feedback model in which all observed edges (u, v) have u active (presuming the algorithm is aware of this fact). This includes the feedback models described above. Fix any policy tree T , and integers $i < j$. We aim to show Eq. (4) from the definition of adaptive submodularity holds, that is

$$f_{\text{avg}}(T_{[j]}^\pi) - f_{\text{avg}}(T_{[j-1]}^\pi) \leq \mathbb{E} \left[f_{\text{avg}}(T_{[i] \cup \{j\}}^\pi) - f_{\text{avg}}(T_{[i]}^\pi) \right].$$

Fix a partial realization Ψ corresponding to the policy tree T 's knowledge after making i selections, and sample a node $v \in V$ from the social network from $\mathcal{D}(T, \Psi, j)$, the distribution on nodes selected by T at layer j conditioned on the realization being consistent with Ψ (i.e., $\Phi \sim \Psi$), as described in §3.

We claim that the marginal contribution of v cannot be larger in $T_{[j]}$ than in $T_{[i] \cup \{j\}}$, despite the fact that when selecting v the former has observed more edges. We couple the distributions on the executions of $T_{[j]}$ and $T_{[i] \cup \{j\}}$ so that we can speak of a common Ψ between them. Let S be the random set of nodes activated by selecting v in $T_{[i] \cup \{j\}}$ conditioned on Ψ , and let S' be the analogous set for $T_{[j]}$. For two random subsets A, B of V , we say A *stochastically dominates* B if for all $U \subseteq V$ we have $\mathbb{P}[U \subseteq B] \leq \mathbb{P}[U \subseteq A]$. Now fix any $B, B' \subseteq V$ such that $B \subseteq B'$, and note that if S stochastically dominates S' then for all Φ

$$\mathbb{E}_{S'}[f(S' \cup B', \Phi) - f(B', \Phi)] \leq \mathbb{E}_S[f(S \cup B, \Phi) - f(B, \Phi)] \quad (10)$$

since $S \mapsto f(S, \Phi)$ is monotone submodular for all realizations Φ . Let B be the set of nodes activated by the first i nodes selected when executing T , and let B' to be the set of nodes activated by the first $j - 1$ selected nodes. Then if we take the expectation of Eq. (10) with respect to sampling $\Phi \sim \Psi$, we get the adaptive submodularity condition for this i, j and T , conditioned on $\Phi \sim \Psi$. Taking an appropriate convex combination of these inequalities over valid choices for Ψ yields the adaptive submodularity condition for our arbitrary choices of i, j and T , and hence proves the overall adaptive submodularity of f .

We now show that S does in fact stochastically dominate S' . Intuitively, S stochastically dominates S' because if an edge (v_1, v_2) has been observed while executing layers in $[i + 1, j - 1]$ then v_1 is already active, and so activating v cannot result in the activation of v_1 , i.e., $v_1 \notin S'$. Moreover if (v_1, v_2) is live, then v_2 is also already active, so $v_2 \notin S'$. On the other hand, if (v_1, v_2) is dead it makes it harder for v to spread influence than if (v_1, v_2) is merely unobserved as yet. More formally, consider any v in the support of $\mathcal{D}(T, \Psi, j)$; here v can depend on the partial realization seen by $T_{[j]}$ just before it makes a selection at layer j , which we denote by Ψ' . Next, fix $\Phi \sim \Psi'$ and consider the graph $(V, A(\Phi))$ of live edges. We argue that if we “remove” the elements of $\text{dom}(\Psi') \setminus \text{dom}(\Psi)$ and their effects (i.e., we deactivate the nodes they influenced), then the set of nodes influenced by playing v can only grow. Let $S(\Phi)$ denote the sets of nodes in influenced by playing v assuming Φ is the true realization and we have already selected $\text{dom}(\Psi)$. Let $S'(\Phi)$ denote the analogous set if we have already selected $\text{dom}(\Psi')$. We aim to prove $S'(\Phi) \subseteq S(\Phi)$. Note $S(\Phi)$ is the set of nodes reachable from v via the live edges $A(\Phi)$, excluding already active nodes (i.e., excluding those reachable from any node in $\text{dom}(\Psi)$ via live edges). The analogous observation holds for $S'(\Phi)$, where the excluded nodes are those reachable from any node in $\text{dom}(\Psi')$ via live edges. Since $\text{dom}(\Psi) \subset \text{dom}(\Psi')$ and the underlying graph $(V, A(\Phi))$ is the same in both cases, we infer $S'(\Phi) \subseteq S(\Phi)$. Hence conditioning on Ψ' , for all $U \subseteq V$ we have

$$\mathbb{P}[U \subseteq S'(\Phi) | \Phi \sim \Psi'] \leq \mathbb{P}[U \subseteq S(\Phi) | \Phi \sim \Psi'].$$

Removing the conditioning on Ψ' by taking the expectation over all Ψ' consistent with Ψ , we infer S stochastically dominates S' , which completes the proof. \blacksquare

7 Application: Active Learning

In pool-based active learning (McCallum & Nigam, 1998), we are given a set of hypotheses H , and a set of unlabeled data points X where each $x \in X$ is independently drawn from some distribution \mathcal{D} . Let L be the set of possible labels. The goal is to adaptively select points to query (i.e., to obtain labels for) until we can output a hypothesis h that will have expected error at most ε with probability $1 - \delta$, for some fixed $\varepsilon, \delta > 0$. That is, if h^* is the target hypothesis (with zero error), and $\text{error}(h) := \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq h^*(x)]$ is the error of h , we require $\mathbb{P}[\text{error}(h) \leq \varepsilon] \geq 1 - \delta$. The latter probability is taken with respect to $\mathcal{D}(X)$; the learned hypothesis h and thus $\text{error}(h)$ depend on it.

In the case of binary labels $L = \{-1, 1\}$, various authors have considered greedy policies which generalize binary search (Garey & Graham, 1974; Loveland, 1985; Arkin et al., 1993; Kosaraju et al., 1999; Dasgupta, 2004; Guillory & Bilmes, 2009; Nowak, 2009). The simplest of these, called *generalized binary search* (GBS) or the *splitting algorithm*, works as follows. Define the *version space* V to be the set of hypotheses consistent with the observed labels (here we assume that there is no label noise). In the worst-case setting, GBS selects a query $x \in X$ that minimizes $|\sum_{h \in V} h(x)|$. In the Bayesian setting we assume we are given a prior p_H over hypotheses; in this case GBS selects a query $x \in X$ that minimizes $|\sum_{h \in V} p_H(h) \cdot h(x)|$. Intuitively these policies myopically attempt to shrink a measure of the version space (i.e., cardinality or the probability mass) as quickly as possible. The former provides an $\mathcal{O}(\log |H|)$ -approximation for the worst-case number of queries (Arkin et al., 1993), and the latter provides an $\mathcal{O}(\log \frac{1}{\min_h p_H(h)})$ -approximation for the expected number of queries (Kosaraju et al., 1999; Dasgupta, 2004) and a natural generalization of GBS obtains the same guarantees with a larger set of labels (Guillory & Bilmes, 2009). Kosaraju *et al.* also point out that running GBS on a modified prior $p'_H(h) \propto \max\{p_H(h), 1/|H|^2 \log |H|\}$ is sufficient to obtain an $\mathcal{O}(\log |H|)$ -approximation.

Viewed from this perspective of the previous sections, shrinking the version space amounts to “covering” all false hypotheses with stochastic sets (i.e., queries), where query x covers all hypotheses that disagree with the target hypothesis h^* at x . That is, x covers $\{h : h(x) \neq h^*(x)\}$. As in §6, these sets may be correlated in complex ways determined by the set of possible hypotheses. Hence the problem is an adaptive stochastic coverage problem in the same vein as Stochastic Submodular Maximization and Adaptive Viral Marketing, except that it is a variant of the Set Cover problem (where we wish to find the cheapest set achieving full coverage) rather than the Maximum Coverage problem (where we want to maximize coverage subject to a budget constraint). We give general results for this adaptive stochastic coverage variant in the extended version of this paper (Golovin & Krause, 2010). We prove the following result, which improves the constant in front of the $\ln(1/\min_h p_H(h))$ to 1. Chakaravarthy et al. (2007) provide a reduction to Set Cover that, with some

parameters slightly tweaked, can be combined with a Set Cover inapproximability result due to Feige (1998) to yield a lower bound of $(1/2 - \epsilon) \ln(1/\min_h p_H(h))$ assuming $\text{NP} \not\subseteq \text{DTIME}(n^{\mathcal{O}(\log \log n)})$.

Theorem 7 *In the Bayesian setting in which there is a prior p_H on a finite set of hypotheses H , the generalized binary search algorithm makes $\text{OPT} \cdot \left(\ln \left(\frac{1}{\min_h p_H(h)} \right) + 1 \right)$ queries in expectation to identify a hypothesis drawn from p_H , where OPT is the minimum expected number of queries made by any policy.*

Due to space limitations, we defer the proof of Theorem 7 to the longer version of this paper. The proof relies on the fact that the reduction in version space is adaptive monotone submodular, which we prove in Proposition 8 below. To define the objective function formally, we first need some notation. Define a realization Φ_h for each hypothesis $h \in H$. The ground set is $E = X$, and the outcomes are binary; we define $O = \{-1, 1\}$ instead of using $\{0, 1\}$ to be consistent with our earlier exposition. For all $h \in H$ we set $\Phi_h \equiv h$, meaning $\Phi_h(x) = h(x)$ for all $x \in X$. Given observed labels $\Psi \subset E \times O$, let $V(\Psi)$ denote the version space, i.e., the set of hypotheses for which $h(x) = \Psi(x)$ for all $x \in \text{dom}(\Psi)$. For a set of hypotheses V , let $p_H(V) := \sum_{h \in V} p_H(h)$ denote their total prior probability. Finally, let $\Psi(S, h) = \{(x, h(x)) : x \in S\}$ be the function with domain S that agrees with h on S . We define the objective function by

$$f(S, \Phi_h) := 1 - p_H(V(\Psi(S, h))) = p_H(\{\Phi : \exists x \in S, \Phi(x) \neq \Phi_h(x)\}) \quad (11)$$

and use $\mathbb{P}[\Phi_h] = p_H(h)$ for all h . Note that identifying h^* as the target hypothesis corresponds to eliminating everything but h^* from the version space.

Proposition 8 *The version space reduction objective is adaptive monotone and adaptive submodular.*

Proof: Adaptive monotonicity is immediate, as additional queries only remove hypotheses from the version space, and never add to it. We establish the adaptive submodularity of f using the characterization in Proposition 3. Each query x eliminates some subset of hypotheses, and as more queries are performed, the subset of hypotheses eliminated by x cannot grow. More formally, consider the expected marginal contribution of x under two partial realizations Ψ, Ψ' where Ψ is a subrealization of Ψ' (i.e., $\Psi \subset \Psi'$), and $x \notin \text{dom}(\Psi')$. Let $\Psi[e/o]$ be the partial realization with domain $\text{dom}(\Psi) \cup \{e\}$ that agrees with Ψ on its domain, and maps e to o . For each $o \in O$, let $a_o := p(V(\Psi[x/o]))$, $b_o := p(V(\Psi'[x/o]))$. Since a hypothesis eliminated from the version space cannot later appear in the version space, we have $a_o \geq b_o$ for all o . Next, note the expected reduction in version space mass (and hence the expected marginal contribution) due to selecting x given partial realization Ψ is

$$\Delta_\Psi(x) = \sum_{o \in O} a_o \cdot \mathbb{P}[\Phi(x) \neq o \mid \Phi \sim \Psi] = \sum_{o \in O} a_o \left(\frac{\sum_{o' \neq o} a_{o'}}{\sum_{o'} a_{o'}} \right) = \frac{\sum_{o \neq o'} a_o a_{o'}}{\sum_{o'} a_{o'}} \quad (12)$$

The corresponding quantity for Ψ' has b_o substituted for a_o in Eq. (12), for each o . Proposition 3 states that proving adaptive submodularity amounts to showing $\Delta_\Psi(x) \geq \Delta_{\Psi'}(x)$. Using Eq. (12), it suffices to show that $\partial\phi/\partial z_o \geq 0$ for each o , where $\phi(\vec{z}) := \left(\sum_{o \neq o'} z_o z_{o'} \right) / \left(\sum_{o'} z_{o'} \right)$ and we assume each $z_o \geq 0$ and $z_o > 0$ for some o . This is because $\partial\phi/\partial z_o \geq 0$ for each o implies that growing the version space in any manner cannot decrease the marginal benefit of query x , and hence shrinking it in any manner cannot increase the marginal benefit of x . The fact $\partial\phi/\partial z_o \geq 0$ for each o can be shown by means of elementary calculus. ■

Extensions to Arbitrary Costs, Multiple Classes, and Approximate Greedy Policies.

This result easily generalizes to handle the multi-class setting (i.e., $|O| \geq 2$), and α -approximate greedy policies, where we lose a factor of α in the approximation factor. As we describe in the extended version of this paper, we can generalize adaptive submodularity to incorporate costs on items, which allows us to extend this result to handle query costs as well. We can therefore recover these extensions of Guillory and Bilmes (2009), while improving the approximation factor for GBS with item costs to $\ln \left(\frac{1}{\min_h p_H(h)} \right) + 1$. Guillory and Bilmes also showed how to extend the technique of Kosaraju et al. (1999) to obtain an $\mathcal{O} \left(\log \left(|H| \frac{\max_x c(x)}{\min_x c(x)} \right) \right)$ -approximation with costs using a greedy policy, which may be combined with our tighter analysis as well to give a similar result with an improved leading constant. Recently, Gupta et al. (2010) showed how to simultaneously remove the dependence on both costs and probabilities from the approximation ratio. Specifically, within the context of studying an adaptive travelling salesman problem they investigated the *Optimal Decision Tree* problem, which is equivalent to the active learning problem we consider here. Using a clever, more complex algorithm than adaptive greedy, they achieve an $\mathcal{O}(\log |H|)$ -approximation in the case of non-uniform costs and general priors.

8 Hardness of Approximation

In this paper, we have developed the notion of adaptive submodularity, which characterizes when certain adaptive stochastic optimization problems are well-behaved in the sense that a simple greedy policy obtains a constant factor or logarithmic factor approximation to the best policy. However, without adaptive submodularity, the Adaptive Stochastic Maximization problem (1) is extremely inapproximable, even with (pointwise) linear objective functions (i.e., those where for each Φ , $f : 2^E \times O^E \rightarrow \mathbb{R}$ is linear in the first argument): We cannot hope to achieve an $\mathcal{O}(|E|^{1-\varepsilon})$ approximation ratio for this problem, unless the polynomial hierarchy collapses to Σ_2^P . Even worse, we can rule out good bicriteria results under the same assumption.

Theorem 9 *In general, for all (possibly non-constant) $\beta \geq 1$, no polynomial time algorithm for Adaptive Stochastic Maximization with a budget of βk items can approximate the reward of an optimal policy with a budget of only k items to within a multiplicative factor of $\mathcal{O}(|E|^{1-\varepsilon}/\beta)$ for any $\varepsilon > 0$, unless $\text{PH} = \Sigma_2^P$. This holds even for pointwise linear f .*

Proof: We construct a hard instance based on the following intuition. We make the algorithm go “treasure hunting”. There is a set of t locations $\{0, 1, \dots, t-1\}$, there is a treasure at one of these locations, and the algorithm gets unit reward if it finds it, and zero reward otherwise. There are m “maps,” each consisting of a cluster of s bits, and each purporting to indicate where the treasure is, and each map is stored in a (weak) secret-sharing way, so that querying few bits of a map reveals nothing about where it says the treasure is. Moreover, all but one of the maps are *fake*, and there is a puzzle indicating which map is the correct one indicating the treasure’s location. Formally, a fake map is one which is probabilistically independent of the location of the treasure, conditioned on the puzzle.

Our instance will have three types of elements, $E = E_T \uplus E_M \uplus E_P$, where $|E_T| = t$ encodes where the treasure is, $|E_M| = ms$ encodes the maps, and $|E_P| = n^3$ encodes the puzzle, where m, t, s and n are specified below. All outcomes are binary, $O = \{0, 1\}$. For all $e \in E_M \cup E_P$, $\mathbb{P}[\Phi(e) = 1] = .5$ independently. The conditional distribution $\mathbb{P}[\Phi(E_T) \mid \Phi(E_M \cup E_P)]$ will be deterministic as specified below. Our objective function f is linear, and defined as follows:

$$f(E', \Phi) = |\{e \in E' \cap E_T : \Phi(e) = 1\}|.$$

We now describe the puzzle, which is to compute $i(P) := (\text{perm}(P) \bmod p) \bmod 2^\ell$ for a suitably sampled random matrix P , and suitable prime p and integer ℓ , where $\text{perm}(P) = \sum_{\sigma \in S_n} \prod_{i=1}^n P_{i\sigma(i)}$ is the permanent of P . We exploit Theorem 1.9 of Feige and Lund (1997) in which they show that if there exist constants $\eta, \delta > 0$ such that a randomized polynomial time algorithm can compute $(\text{perm}(P) \bmod p) \bmod 2^\ell$ correctly with probability $2^{-\ell}(1 + 1/n^\eta)$, where P is drawn uniformly at random from $\{0, 1, 2, \dots, p-1\}^{n \times n}$, p is any prime superpolynomial in n , and $\ell \leq p(\frac{1}{2} - \delta)$, then $\text{PH} = \text{AM} = \Sigma_2^P$. To encode the puzzle, we fix a prime $p \in [2^{n-2}, 2^{n-1}]$ and use the n^3 bits of $\Phi(E_P)$ to sample $P = P(\Phi)$ (nearly) uniformly at random from $\{0, 1, 2, \dots, p-1\}^{n \times n}$ as follows. For a matrix $P \in \mathbb{Z}^{n \times n}$, we let $\text{rep}(P) := \sum_{i,j} P_{ij} \cdot p^{(i-1)n+(j-1)}$ define a base p representation of P . Note $\text{rep}(\cdot)$ is one-to-one for $n \times n$ matrices with entries in \mathbb{Z}_p , so we can define its inverse $\text{rep}^{-1}(\cdot)$. The encoding $P(\Phi)$ interprets the bits $\Phi(E_P)$ as an integer x in $[2^{n^3}]$, and computes $y = x \bmod (p^{n^2})$. If $x \leq \lfloor 2^{n^3}/p^{n^2} \rfloor p^{n^2}$, then $P = \text{rep}^{-1}(y)$. Otherwise, P is the all zero matrix. This latter event occurs with probability at most $p^{n^2}/2^{n^3} \leq 2^{-n^2}$, and in this case we simply suppose the algorithm under consideration finds the treasure and so gets unit reward. This adds 2^{-n^2} to its expected reward. So let us assume from now on that P is drawn uniformly at random.

Next we consider the maps. Partition $E_M = \uplus_{i=1}^m M_i$ into m maps M_i , each consisting of s items. For each map M_i , partition its items into $s/\log_2 t$ groups of $\log_2 t$ bits each, and let $v_i \in \{0, 1, \dots, t-1\}$ be the XOR of these groups of bits. We say M_i *points to* v_i as the location of the treasure. A priori, each v_i is uniformly distributed in $\{0, \dots, t-1\}$. For a particular realization of $\Phi(E_P \cup E_M)$, define $v(\Phi) := v_{i(P(\Phi))}$. We set $v(\Phi)$ to be the location of the treasure under realization Φ , i.e., we label $E_T = \{e_0, e_1, \dots, e_{t-1}\}$ and ensure $\Phi(e_j) = 1$ if $j = v_{i(P(\Phi))}$, and $\Phi(e) = 0$ for all other $e \in E_T$. Note the random variable $v = v(\Phi)$ is distributed uniformly at random in $\{0, 1, \dots, t-1\}$. Note that this still holds if we condition on the realizations of any set of $s/\log_2 t - 1$ items in a map.

Now consider the optimal policy with a budget of $k = n^3 + s + 1$ items to pick. Clearly, its reward can be at most 1. However, given a budget of k , a computationally unconstrained policy can exhaustively sample E_P , solve the puzzle (i.e., compute $i(P)$), read the correct map (i.e., exhaustively sample $M_{i(P)}$), decode the map (i.e., compute $v = v_{i(P)}$), and get the treasure (i.e., pick e_v) thereby obtaining a reward of one.

Now we give an upper bound on the expected reward R of any randomized polynomial time algorithm \mathcal{A} with a budget of βk items, assuming $\Sigma_2^P \neq \text{PH}$. Fix a small constant $\gamma > 0$, and set $s = n^3$ and $m = t = n^{1/\gamma}$. We suppose we give \mathcal{A} the realizations $\Phi(E_M)$ for free. We also replace its budget of

βk items with a budget of βk specifically for map items in E_M and an additional budget of βk specifically for the treasure locations in E_T . Obviously, this can only help it. As noted, if it selects less than $s/\log_2 t$ bits from the map $M_{i(P)}$ indicated by P , the distribution over $v_{i(P)}$ conditioned on those realizations is still uniform. Of course, knowledge of v_i for $i \neq i(P)$ is useless for getting reward. Hence \mathcal{A} can try at most $\beta k \log_2(t)/s = o(\beta k)$ maps in an attempt to find $M_{i(P)}$. Note that if we have a randomized algorithm which given a random P drawn from $\{0, 1, 2, \dots, p-1\}^{n \times n}$ always outputs a set S of integers of size α such that $\mathbb{P}[i(P) \in S] \geq q$, then we can use it to construct a randomized algorithm that, given P , outputs an integer x such that $\mathbb{P}[i(P) = x] \geq q/\alpha$, simply by running the first algorithm and then selecting a random element of S . If \mathcal{A} does not find $M_{i(P)}$, the distribution on the treasure's location is uniform given its knowledge. Hence it's budget of βk treasure locations can only earn it expected reward at most $\beta k/t$. Armed with these observations and Theorem 1.9 of Feige and Lund (1997) and our complexity theoretic assumptions, we infer $\mathbb{E}[R] \leq o(\beta k) \cdot 2^{-\ell}(1 + 1/n^n) + \beta k/t + 2^{-n^2}$. Since $s = n^3$ and $m = t = n^{1/\gamma}$ and $\gamma = \Theta(1)$ and $\eta = 1$ and $\ell = \log_2 m$ and $k = n^3 + s + 1 = 2n^3 + 1$, we have

$$\mathbb{E}[R] \leq \frac{\beta k}{t} (1 + o(1)) = 2\beta n^{3-1/\gamma}(1 + o(1)).$$

Next note that $|E| = t + ms + n^3 = n^{3+1/\gamma}(1 + o(1))$. Straightforward algebra shows that in order to ensure $\mathbb{E}[R] = o(\beta/|E|^{1-\varepsilon})$, it suffices to choose $\gamma \leq \varepsilon/6$. Thus, under our complexity theoretic assumptions, any polynomial time randomized algorithm \mathcal{A} with budget βk achieves at most $o(\beta/|E|^{1-\varepsilon})$ of the value obtained by the optimal policy with budget k , so the approximation ratio is $\omega(|E|^{1-\varepsilon}/\beta)$. ■

9 Conclusions

In this paper, we introduced the concept of *adaptive submodularity*, generalizing submodular set functions to adaptive policies. Our generalization is based on a natural adaptive analog of the diminishing returns property well understood for set functions. In the special case of deterministic distributions, adaptive submodularity reduces to the classical notion of submodular set functions. We proved that guarantees carried by the non-adaptive greedy algorithm for submodular set functions generalize to a natural adaptive greedy algorithm in the case of adaptive submodular functions. We illustrated the usefulness of the concept by giving several examples of adaptive submodular objectives arising in diverse applications including sensor placement, viral marketing and pool-based active learning. Proving adaptive submodularity for these problems allowed us to recover existing results in these applications as special cases and leads to natural generalizations. We believe that our results provide an interesting step in the direction of exploiting structure to solve complex stochastic optimization problems under partial observability.

Acknowledgments

This research was partially supported by ONR grant N00014-09-1-1044, NSF grant CNS-0932392, NSF grant IIS-0953413, a gift by Microsoft Corporation, an Okawa Foundation Research Grant, and by the Caltech Center for the Mathematics of Information.

References

- Arkin, E. M., Meijer, H., Mitchell, J. S. B., Rappaport, D., & Skiena, S. S. (1993). Decision trees for geometric models. *Proc. of Symposium on Computational Geometry* (pp. 369–378). New York, NY, USA: ACM.
- Asadpour, A., Nazerzadeh, H., & Saberi, A. (2008). Stochastic submodular maximization. *WINE '08: Proc. of the 4th International Workshop on Internet and Network Economics* (pp. 477–489). Berlin, Heidelberg: Springer-Verlag.
- Chakaravarthy, V. T., Pandit, V., Roy, S., Awasthi, P., & Mohania, M. (2007). Decision trees for entity identification: Approximation algorithms and hardness results. *In Proceedings of the ACM-SIGMOD Symposium on Principles of Database Systems*.
- Dasgupta, S. (2004). Analysis of a greedy active learning strategy. *NIPS: Advances in Neural Information Processing Systems* (pp. 337–344). MIT Press.
- Feige, U. (1998). A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45, 634 – 652.
- Feige, U., & Lund, C. (1997). On the hardness of computing the permanent of random matrices. *Computational Complexity*, 6, 101–132.

- Fujishige, S. (1991). *Submodular functions and optimization*. No. 47. North Holland, Amsterdam: Annals of Discrete Mathematics. 2nd edition.
- Garey, M. R., & Graham, R. L. (1974). Performance bounds on the splitting algorithm for binary testing. *Acta Inf.*, 3, 347–355.
- Goemans, M. X., & Vondrák, J. (2006). Stochastic covering and adaptivity. *LATIN* (pp. 532–543). Springer.
- Golovin, D., & Krause, A. (2010). Adaptive submodularity: A new approach to active learning and stochastic optimization. *CoRR*, *abs/1003.3967*.
- Goundan, P. R., & Schulz, A. S. (2007). *Revisiting the greedy approach to submodular set function maximization* (Technical Report). Massachusetts Institute of Technology.
- Guillory, A., & Bilmes, J. (2009). Average-case active learning with costs. *The 20th International Conference on Algorithmic Learning Theory*. University of Porto, Portugal.
- Guillory, A., & Bilmes, J. (2010). Interactive submodular set cover. *To appear in Proc. International Conference on Machine Learning (ICML)*.
- Gupta, A., Krishnaswamy, R., Nagarajan, V., & Ravi, R. (2010). Approximation algorithms for optimal decision trees and adaptive tsp problems. *To appear in Proc. International Colloquium on Automata, Languages and Programming (ICALP)*.
- Kempe, D., Kleinberg, J., & Tardos, E. (2003). Maximizing the spread of influence through a social network. *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 137–146). New York, NY, USA: ACM.
- Kosaraju, S. R., Przytycka, T. M., & Borgstrom, R. S. (1999). On an optimal split tree problem. *Proc. of the 6th Intl. Workshop on Algorithms and Data Structures* (pp. 157–168). London, UK: Springer-Verlag.
- Krause, A., & Guestrin, C. (2007). Near-optimal observation selection using submodular functions. *AAAI Nectar track* (pp. 1650–1654).
- Loveland, D. W. (1985). Performance bounds for binary testing with arbitrary weights. *Acta Inf.*, 22, 101–114.
- McCallum, A., & Nigam, K. (1998). Employing EM and pool-based active learning for text classification. *ICML* (pp. 350–358). Morgan Kaufmann.
- Minoux, M. (1978). Accelerated greedy algorithms for maximizing submodular set functions. *Proc. of the 8th IFIP Conference on Optimization Techniques* (pp. 234–243). Springer.
- Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14, 265–294.
- Nowak, R. (2009). Noisy generalized binary search. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams and A. Culotta (Eds.), *Advances in neural information processing systems 22*, 1366–1374.
- Schrijver, A. (2003). *Combinatorial optimization : polyhedra and efficiency*. Volume B, Part IV, Chapters 39–49. Springer.

Robust Selective Sampling from Single and Multiple Teachers

Ofer Dekel

Microsoft Research
oferd@microsoft.com

Claudio Gentile

DICOM, Università dell’Insubria
claudio.gentile@uninsubria.it

Karthik Sridharan

TTI-Chicago
karthik@ttic.edu

Abstract

We present a new online learning algorithm in the selective sampling framework, where labels must be actively queried before they are revealed. We prove bounds on the regret of our algorithm and on the number of labels it queries when faced with an adaptive adversarial strategy of generating the instances. Our bounds both generalize and strictly improve over previous bounds in similar settings. Using a simple online-to-batch conversion technique, our selective sampling algorithm can be converted into a statistical (pool-based) active learning algorithm. We extend our algorithm and analysis to the multiple-teacher setting, where the algorithm can choose which subset of teachers to query for each label.

1 Introduction

A *selective sampling* algorithm (Cohn et al., 1990; Freund et al., 1997) is an online learning algorithm that actively decides which labels to query. More precisely, learning takes place in a sequence of rounds. On round t , the online learner receives an instance $\mathbf{x}_t \in \mathbb{R}^d$ and predicts a binary label $\hat{y}_t \in \{-1, +1\}$. Then, the learner decides whether or not to *query* the true label y_t associated with \mathbf{x}_t . If the label is queried, the learner incurs a unit cost and uses the label to improve his future predictions. If the label is not queried, the learner never knows whether his prediction was correct. Nevertheless, the accuracy of the learner is evaluated on both queried and unqueried instances. We say that a selective sampling algorithm is *robust* if it works even when the instance sequence $\mathbf{x}_1, \mathbf{x}_2, \dots$ is generated by an *adaptive adversary*. Robustness thereby implies a high level of adaptation to the learning environment.

Inspired by known online ridge regression algorithms (e.g., (Hoerl & Kennard, 1970; Lai & Wei, 1982; Vovk, 2001; Azoury & Warmuth, 2001; Cesa-Bianchi et al., 2003; Cesa-Bianchi et al., 2005; Li et al., 2008; Strehl & Littman, 2008; Cavallanti et al., 2009; Cesa-Bianchi et al., 2009)), we begin by presenting a new robust selective sampling algorithm within the label-noise setting considered in (Cavallanti et al., 2009; Cesa-Bianchi et al., 2009; Strehl & Littman, 2008). We measure the predictive accuracy of our learner using the game-theoretic notion of *regret* (formally defined below) and prove formal bounds on this quantity. We also prove bounds on the number of queries issued by the learner. Our bounds are strictly better than the best available bounds in the robust selective sampling setting, and can be shown to be optimal with respect to certain parameters. A detailed comparison of our results with the results of the predominant previous papers on this topic (Cesa-Bianchi et al., 2006; Strehl & Littman, 2008; Cesa-Bianchi et al., 2009) is given in Section 2.5, after our results are presented.

Selective sampling can be viewed as an online-learning variant of active learning. The literature on active learning is vast, and we can hardly do it justice here. Recent papers on active learning include (Balcan et al., 2006; Balcan et al., 2007; Castro & Nowak, 2008; Dasgupta et al., 2008; Dasgupta et al., 2005; Hanneke, 2007; Hanneke, 2009). All of these papers consider the case where instances are drawn i.i.d. from a fixed distribution (either known or unknown). As a by-product of our adversarial analysis, we also obtain a tight regret bound in the case where the instances \mathbf{x}_t are generated i.i.d. according to a fixed and unknown distribution. Moreover, using a simple online-to-batch conversion technique, our online learner becomes a randomized statistical pool-based active-learning algorithm, with a high-probability risk bound.

In the second part of this paper, we extend our algorithm and analysis to the case where the learner has access to multiple teachers, each one with a different area of expertise and a different level of overall competence. In other words, the learner is free to query any subset of teachers and each teacher is capable of providing accurate labels only within some subset of the instance space. The learner is not given any information on the expertise region of each teacher, and must infer this information directly from the labels. Roughly speaking, the goal of the learner is to perform as well as each teacher in his respective area of

expertise. We first present an online learner that either queries all of the teachers or does not query any teacher. We then enhance this learner to query only those teachers it believes to be experts on \mathbf{x}_t .

The general aim of this line of research is to provide algorithms of practical utility for which we can also prove formal performance guarantees. The motivation behind selective sampling is the same as the motivation behind any active learning algorithm: human-generated labels are expensive and therefore we only want labels that improve our ability to make accurate predictions. Our work within the multiple teacher setting is motivated by an Internet search company that uses online learning techniques to determine the results of its search engine. More concretely, the instance \mathbf{x}_t represents the pairing of a search-engine query with a candidate web page; the goal of the online learner is to determine whether or not this pair constitutes a good match. The company employs human teachers to provide the correct answer for any instance. Clearly, there is no way to manually label the millions of daily search engine queries, and some intelligent mechanism of choosing which instances to label is required. Each teacher provides labels of different quality in different regions of the instance space. To make accurate predictions, the learner must figure out which teachers to trust for each instance.

A learning framework sharing similar motivations to ours is the proactive learning setting (Donmez & Carbonell, 2008; Yang & Carbonell, 2009a), where the learner has access to teachers of different quality, with associated costs per label. Yang and Carbonell (2009b) presents a theoretical analysis of proactive learning, however, this analysis relies on the strong assumption that each teacher gives the correct label most of the time. We make no such assumption in our analysis. Moreover, our setting supports the realistic scenario where each teacher has a very narrow area of expertise and gives useless labels outside of this area.

2 The Single Teacher Case

In this section, we focus on the standard online selective sampling setting, where the learner has to learn an accurate predictor while determining whether or not to query the label of each instance it observes. In this setting, the learner has no control over where the label comes from.

2.1 Preliminaries and Notation

As mentioned above, on round t of the online learning process, the learner receives input $\mathbf{x}_t \in \mathbb{R}^d$, predicts $\hat{y}_t \in \{-1, +1\}$, and chooses whether or not to query the correct label $y_t \in \{-1, +1\}$. We set $Z_t = 1$ if a query is issued and $Z_t = 0$ otherwise. The only assumption we make on the process that generates \mathbf{x}_t is that $\|\mathbf{x}_t\| \leq 1$; for all we know instances may be generated by an *adaptive* adversary. Note that most of the previous work on this topic makes stronger assumptions on the process that generates \mathbf{x}_t , leading to a less general setting. As for the labels, we adopt the standard stochastic linear noise¹ model for this problem (Cesa-Bianchi et al., 2003; Cavallanti et al., 2009; Cesa-Bianchi et al., 2009; Strehl & Littman, 2008) and assume that each $y_t \in \{-1, +1\}$ is sampled according to the law $P(y_t = 1 | \mathbf{x}_t) = (1 + \mathbf{u}^\top \mathbf{x}_t)/2$, where $\mathbf{u} \in \mathbb{R}^d$ is a fixed but unknown vector with $\|\mathbf{u}\| \leq 1$. Note that under this setup, $\mathbb{E}[y_t | \mathbf{x}_t] = \mathbf{u}^\top \mathbf{x}_t$, and we denote the latter by Δ_t . The learner uses hyperplanes to predict the label on each round. That is, on round t the learner predicts $\hat{y}_t = \text{sign}(\hat{\Delta}_t)$ where $\hat{\Delta}_t = \mathbf{w}_{t-1}^\top \mathbf{x}_t$. Let P_t denote the conditional probability $\mathbb{P}(\cdot | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t, y_1, \dots, y_{t-1})$. We evaluate the accuracy of the learner's predictions using its cumulative *regret*, defined as

$$R_T = \sum_{t=1}^T \left(P_t(y_t \hat{\Delta}_t < 0) - P_t(y_t \Delta_t < 0) \right) .$$

Additionally, we are interested in the number of queries issued by the learner $N_T = \sum_{t=1}^T Z_t$. Our goal is to simultaneously bound the cumulative regret R_T and the number of queries N_T with high probability over the random draw of labels.

2.2 Algorithm

The single teacher algorithm is a margin-based selective sampling procedure. The algorithm ‘‘Selective Sampler’’ (Algorithm 1) depends on a confidence parameter $\delta \in (0, 1]$. As in known online ridge-regression-like algorithms (e.g., (Hoerl & Kennard, 1970; Vovk, 2001; Azoury & Warmuth, 2001; Cesa-Bianchi et al., 2003; Cesa-Bianchi et al., 2005; Li et al., 2008; Strehl & Littman, 2008; Cavallanti et al., 2009; Cesa-Bianchi et al., 2009)), our algorithm maintains a weight vector \mathbf{w}_t (initialized as $\mathbf{w}_0 = \mathbf{0}$) and a data correlation matrix A_t (initialized as $A_0 = I$). After receiving \mathbf{x}_t and predicting $\hat{y}_t = \text{sign}(\hat{\Delta}_t)$, the algorithm computes an adaptive data-dependent threshold θ_t , defined as

$$\theta_t^2 = \mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t \left(1 + 4 \sum_{i=1}^{t-1} Z_i r_i + 36 \log(t/\delta) \right) ,$$

¹The noise model we are adopting here not only can be made more general (i.e., highly nonlinear) by the use of kernel functions (see Section 2.2), but has also undergone a rather thorough experimental validation on real-world data (Cavallanti et al., 2009; Cesa-Bianchi et al., 2009).

where $r_i = \mathbf{x}_i^\top A_i^{-1} \mathbf{x}_i$. The definition of θ_t derives from our analysis, and can be interpreted as the algorithm's uncertainty in its own predictions. More precisely, the learner believes that $|\hat{\Delta}_t - \Delta_t| \leq \theta_t$. A query is issued only if² $|\hat{\Delta}_t| \leq \theta_t$, or in other words, when the algorithm is unsure about the sign of Δ_t .

It is important to stress how θ_t depends on the three terms $\sum_{i=1}^{t-1} Z_i r_i$, $\log(t/\delta)$, and $\mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t$. We can prove that $\sum_{i=1}^t Z_i r_i$ grows only logarithmically with the number of queries N_t , and obviously $\log(t/\delta)$ grows logarithmically with t . The behavior of the third term, $\mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t$, depends on the relationship between the current instance \mathbf{x}_t and the previous instances. If \mathbf{x}_t lies along the directions spanned by the previous instances then we can show that $\mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t$ tends to shrink as $1/N_t$. As a result, the threshold θ_t is on the order of $\log(t/\delta)/N_t$ and the algorithm keeps querying labels at a slow logarithmic rate. On the other hand, if the adversary chooses \mathbf{x}_t to lie outside of the subspace spanned by the previous examples, then the term $\mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t$ causes θ_t to be large, and the algorithm is more likely to issue a query. Overall, to ensure a small uncertainty threshold θ_t over all input directions determined by the adversarial choice of \mathbf{x}_t , the algorithm must query on the order of $\log(t)$ labels for each such direction in the instance space.

If the label is not queried, ($Z_t = 0$) then the algorithm does not update its internal state. If the label is queried ($Z_t = 1$), then the algorithm computes the intermediate vector \mathbf{w}'_{t-1} in such a way that $\hat{\Delta}'_t = \mathbf{w}'_{t-1}^\top \mathbf{x}_t$ is at most one in magnitude. Observe that $\hat{\Delta}_t$ and $\hat{\Delta}'_t$ have the same sign and only their magnitudes can differ. In particular, it holds that

$$\hat{\Delta}'_t = \begin{cases} \text{sgn}(\hat{\Delta}_t) & \text{if } |\hat{\Delta}_t| > 1 \\ \hat{\Delta}_t & \text{otherwise} \end{cases} .$$

Next, the algorithm defines the new vector \mathbf{w}_t so that $A_t \mathbf{w}_t$ undergoes an additive update, where A_t is a rank-one adjustment of A_{t-1} .

It is not hard to show that this algorithm has a quadratic running time per round, where quadratic means $O(d^2)$ if it is run in primal form, and $O(N_t^2)$ if it is run in dual form (i.e., in a reproducing kernel Hilbert space). In the dual case, since the algorithm updates only when $Z_t = 1$, the number of labels N_t corresponds to the number of support vectors used to define the current hypothesis.

2.3 Analysis

We now prove formal guarantees on the regret of the algorithm and the number of labels it queries. Some details are omitted due to space constraints, and the interested reader is referred to (Dekel et al., 2010) for a more complete analysis. Following (Cesa-Bianchi et al., 2009), the bounds we give depend on how many of the (adversarially chosen) inputs \mathbf{x}_t are close to being complete noise. To capture this dependence, for any $\epsilon > 0$, define

$$T_\epsilon = \sum_{t=1}^T \mathbb{1}\{|\Delta_t| \leq \epsilon\} . \quad (1)$$

Note that if $|\Delta_t| \leq \epsilon$ then $P_t(y_t = 1) \in [1/2 + \epsilon, 1/2 - \epsilon]$. In short, T_ϵ is a ‘‘hardness’’ parameter which is essentially controlled by the adversary. This need not be the case when data is i.i.d. (see Section 2.4). The following theorem is the main result of this section, and is stated so as to emphasize both the data-dependent and the time-dependent aspects of our bounds.

Theorem 1 *Assume that Selective Sampler is run with confidence parameter $\delta \in (0, 1]$. Then with probability at least $1 - \delta$ it holds that for all $T > 0$ that*

$$\begin{aligned} R_T &\leq \inf_{\epsilon > 0} \left\{ \epsilon T_\epsilon + \frac{2 + 8 \log |A_T| + 144 \log(T/\delta)}{\epsilon} \right\} = \inf_{\epsilon > 0} \left\{ \epsilon T_\epsilon + O\left(\frac{d \log T + \log(T/\delta)}{\epsilon}\right) \right\} \\ N_T &\leq \inf_{\epsilon > 0} \left\{ T_\epsilon + O\left(\frac{\log |A_T| \log(T/\delta) + \log^2 |A_T|}{\epsilon^2}\right) \right\} = \inf_{\epsilon > 0} \left\{ T_\epsilon + O\left(\frac{d^2 \log^2(T/\delta)}{\epsilon^2}\right) \right\} , \end{aligned}$$

where $|A_T|$ is the determinant of the matrix A_T .

As in (Cesa-Bianchi et al., 2009) it is easy to see that the algorithm can also be run in an infinite dimensional reproducing kernel Hilbert space. In this case, the dimension d in the bounds above is replaced by a quantity that depends on the spectrum of the data's Gram matrix.

The proof of Theorem 1 splits into a series of lemmas. For every $T > 0$ and $\epsilon > 0$, we define

$$U_{T,\epsilon} = \sum_{t=1}^T \bar{Z}_t \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0\} \quad \text{and} \quad Q_{T,\epsilon} = \sum_{t=1}^T Z_t \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 > \epsilon^2\} |\Delta_t| ,$$

²This is denoted by $Z_t = \mathbb{1}\{|\hat{\Delta}_t| \leq \theta_t\}$ in the algorithm's pseudocode. Here and throughout $\mathbb{1}\{\cdot\}$ denotes the indicator function.

Algorithm 1: Selective Sampler

input confidence level $\delta \in (0, 1]$
initialize $\mathbf{w}_0 = \mathbf{0}$, $A_0 = I$
for $t = 1, 2, \dots$
 receive $\mathbf{x}_t \in \mathbb{R}^d$: $\|\mathbf{x}_t\| \leq 1$, and set $\hat{\Delta}_t = \mathbf{w}_{t-1}^\top \mathbf{x}_t$
 predict $\hat{y}_t = \text{sgn}(\hat{\Delta}_t) \in \{-1, +1\}$
 $\theta_t^2 = \mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t \left(1 + 4 \sum_{i=1}^{t-1} Z_i r_i + 36 \log(t/\delta)\right)$
 $Z_t = \mathbb{1} \{ \hat{\Delta}_t^2 \leq \theta_t^2 \} \in \{0, 1\}$
 if $Z_t = 1$
 query $y_t \in \{-1, +1\}$
 $\mathbf{w}'_{t-1} = \begin{cases} \mathbf{w}_{t-1} - \left(\frac{|\hat{\Delta}_t| - 1}{\mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t}\right) A_{t-1}^{-1} \mathbf{x}_t & \text{if } |\hat{\Delta}_t| > 1 \\ \mathbf{w}_{t-1} & \text{otherwise} \end{cases}$
 $A_t = A_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top$, $r_t = \mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t$, $\mathbf{w}_t = A_t^{-1} (A_{t-1} \mathbf{w}'_{t-1} + y_t \mathbf{x}_t)$
 else
 $A_t = A_{t-1}$, $\mathbf{w}_t = \mathbf{w}_{t-1}$, $r_t = 0$

where $\bar{Z}_t = 1 - Z_t$. In the above, $U_{T,\epsilon}$ deals with rounds where the algorithm does not make a query, while $Q_{T,\epsilon}$ deals with rounds where the algorithm does make a query. The proof exploits the potential-based method (e.g., (Cesa-Bianchi & Lugosi, 2006)) for online ridge-regression-like algorithms introduced in (Azoury & Warmuth, 2001). See also (Hazan et al., 2006; Dani et al., 2008) for a similar use in different contexts. The potential function we use is the (quadratic) Bregman divergence $d_t(\mathbf{u}, \mathbf{w}) = \frac{1}{2} (\mathbf{u} - \mathbf{w})^\top A_t (\mathbf{u} - \mathbf{w})$, where A_t is the matrix computed by Selective Sampler at time t . The proof structure is as follows. First, Lemma 2 below decomposes the regret R_T into 3 parts: $R_T \leq \epsilon T_\epsilon + U_{T,\epsilon} + Q_{T,\epsilon}$. The bound on $U_{T,\epsilon}$ is given by Lemma 3. For the bound on $Q_{T,\epsilon}$ and the bound on the number of queries N_T , we use Lemmas 4 and 5, respectively. However, both of these lemmas require that $(\Delta_t - \hat{\Delta}_t)^2 \leq \theta_t^2$ for all t . This assumption is taken care of by the subsequent Lemma 6. Since ϵ is a positive free parameter, we can take the infimum over $\epsilon > 0$ to get the required results.

Lemma 2 For any $\epsilon > 0$ it holds that $R_T \leq \epsilon T_\epsilon + U_{T,\epsilon} + Q_{T,\epsilon}$.

Proof: We have

$$\begin{aligned} P_t(\hat{\Delta}_t y_t < 0) - P_t(\Delta_t y_t < 0) &\leq \mathbb{1}\{\hat{\Delta}_t \Delta_t \leq 0\} |2P_t(y_t = 1) - 1| = \mathbb{1}\{\hat{\Delta}_t \Delta_t \leq 0\} |\Delta_t| \\ &= \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 \leq \epsilon^2\} |\Delta_t| + \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 > \epsilon^2\} |\Delta_t| \\ &\leq \epsilon \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 \leq \epsilon^2\} + \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 > \epsilon^2\} |\Delta_t| \\ &\leq \epsilon \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 \leq \epsilon^2\} + \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 > \epsilon^2, Z_t = 0\} |\Delta_t| \\ &\quad + \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 > \epsilon^2, Z_t = 1\} |\Delta_t| \\ &\leq \epsilon \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 \leq \epsilon^2\} + \bar{Z}_t \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 > \epsilon^2\} + Z_t \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 > \epsilon^2\} |\Delta_t|. \end{aligned} \tag{2}$$

Summing over $t = 1 \dots T$ completes the proof. \blacksquare

Lemma 3 For any $\epsilon > 0$ and $T > 0$, with probability at least $1 - \delta$ it holds that

$$Q_{T,\epsilon} \leq \frac{2 + 8 \log |A_T| + 144 \log(T/\delta)}{\epsilon} = O\left(\frac{d \log T + \log(T/\delta)}{\epsilon}\right).$$

Proof Sketch: Using the fact that $\text{sgn}(\hat{\Delta}_t) = \text{sgn}(\hat{\Delta}'_t)$ and the inequality $\mathbb{1}\{\Delta_t^2 > \epsilon^2\} \leq |\Delta_t|/\epsilon$, we upper bound $Q_{T,\epsilon} \leq \frac{1}{\epsilon} \sum_{t=1}^T Z_t \mathbb{1}\{\hat{\Delta}'_t \Delta_t < 0\} \Delta_t^2$. Moreover, $\hat{\Delta}'_t \Delta_t < 0$ implies $\Delta_t^2 \leq (\Delta_t - \hat{\Delta}'_t)^2$ and therefore $Q_{T,\epsilon} \leq \frac{1}{\epsilon} \sum_{t=1}^T Z_t (\Delta_t - \hat{\Delta}'_t)^2$. Next, we define $M_t = Z_t (\Delta_t - y_t) (\Delta_t - \hat{\Delta}'_t)$ and note that

$$\sum_{t=1}^T M_t = \frac{1}{2} \sum_{t=1}^T Z_t (\Delta_t - \hat{\Delta}'_t)^2 - \frac{1}{2} \sum_{t=1}^T Z_t \left((y_t - \hat{\Delta}'_t)^2 - (y_t - \Delta_t)^2 \right).$$

We also note that $(M_t)_{t=1}^T$ is a martingale difference sequence with $|M_i| \leq 4$. Using martingale tail bounds from (Kakade & Tewari, 2008), we obtain a high probability upper bound on $\sum_{t=1}^T M_t$, which implies that

$$\frac{1}{\epsilon} \sum_{t=1}^T Z_t (\Delta_t - \hat{\Delta}'_t)^2 \leq \frac{2}{\epsilon} \sum_{t=1}^T Z_t \left((y_t - \hat{\Delta}'_t)^2 - (y_t - \Delta_t)^2 \right) + \frac{144}{\epsilon} \log(T/\delta) .$$

Finally, we use techniques from (Azoury & Warmuth, 2001) to upper bound the above by

$$\frac{4}{\epsilon} \sum_{t=1}^T Z_t (d_{t-1}(\mathbf{u}, \mathbf{w}'_{t-1}) - d_t(\mathbf{u}, \mathbf{w}'_t) + 2 \log |A_t| - 2 \log |A_{t-1}|) + \frac{144}{\epsilon} \log(T/\delta) .$$

We are left with a telescoping sum that collapses to the desired upper bound. \blacksquare

Lemma 4 Assume that $(\Delta_t - \hat{\Delta}_t)^2 \leq \theta_t^2$ holds for all t . Then, for any $\epsilon > 0$, we have $U_{T,\epsilon} = 0$

Proof: We rewrite our assumption $(\Delta_t - \hat{\Delta}_t)^2 \leq \theta_t^2$ as $\Delta_t \hat{\Delta}_t \geq \frac{\hat{\Delta}_t^2 + \Delta_t^2 - \theta_t^2}{2} \geq \frac{\hat{\Delta}_t^2 - \theta_t^2}{2}$. However, if $\bar{Z}_t = 1$, then $\hat{\Delta}_t^2 > \theta_t^2$ and so $\Delta_t \hat{\Delta}_t \geq 0$. Hence, under the above assumption, we can guarantee that for any t , $\bar{Z}_t \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0\} = 0$, thereby implying $U_{T,\epsilon} = \sum_{t=1}^T \bar{Z}_t \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0, \Delta_t^2 > \epsilon^2\} = 0$. \blacksquare

Lemma 5 Assume that $(\Delta_t - \hat{\Delta}_t)^2 \leq \theta_t^2$ holds for all t . Then, for any $\epsilon > 0$, we have

$$N_T \leq T_\epsilon + O\left(\frac{\log |A_T| \log(T/\delta) + \log^2 |A_T|}{\epsilon^2}\right) = T_\epsilon + O\left(\frac{d^2 \log^2(T/\delta)}{\epsilon^2}\right) .$$

Proof Sketch: Define $\beta_t = \frac{\epsilon^2 \mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t}{8 r_t}$ and rewrite

$$Z_t = Z_t \mathbb{1}\{\theta_t^2 < \beta_t\} + Z_t \mathbb{1}\{\theta_t^2 \geq \beta_t\} = \mathbb{1}\{\hat{\Delta}_t^2 \leq \theta_t^2, \theta_t^2 < \beta_t\} + Z_t \mathbb{1}\{\theta_t^2 \geq \beta_t\} .$$

We begin by dealing with the first term on the right-hand side above. Our assumption implies that whenever $\hat{\Delta}_t^2 \leq \theta_t^2$ it also holds that $\Delta_t^2 \leq 4\theta_t^2$. Hence we can upper bound the first term by $\mathbb{1}\{\Delta_t^2 \leq 4\theta_t^2, \theta_t^2 < \beta_t\}$. Using technical results from (Azoury & Warmuth, 2001), we have that $\beta_t \leq \epsilon^2/4$, and we can further upper bound $\mathbb{1}\{\Delta_t^2 \leq 4\theta_t^2, \theta_t^2 < \beta_t\} \leq \mathbb{1}\{\Delta_t^2 \leq \epsilon^2\}$. Summing over t gives

$$N_T = \sum_{t=1}^T Z_t \leq T_\epsilon + \sum_{t=1}^T Z_t \mathbb{1}\{\theta_t^2 \geq \beta_t\} .$$

Next, we use the definitions of Z_t, β_t and θ_t to get,

$$\begin{aligned} \sum_{t=1}^T Z_t \mathbb{1}\{\theta_t^2 \geq \beta_t\} &= \sum_{t=1}^T Z_t \mathbb{1}\left\{8 r_t (1 + 4 \sum_{i=1}^{t-1} Z_i r_i + 36 \log(t/\delta)) \geq \epsilon^2\right\} \\ &\leq \frac{8}{\epsilon^2} \sum_{t=1}^T Z_t r_t (1 + 4 \sum_{i=1}^{t-1} Z_i r_i + 36 \log(t/\delta)) . \end{aligned}$$

Once again relying on results from (Azoury & Warmuth, 2001), we have that $Z_i r_i \leq \log |A_i| - \log |A_{i-1}|$ and the above can be upper bounded by

$$\frac{8}{\epsilon^2} (1 + 36 \log(T/\delta)) \log |A_T| + \frac{16}{\epsilon^2} \log^2 |A_T| .$$

This concludes the proof. \blacksquare

Lemma 6 If Selective Sampler is run with confidence parameter $\delta \in (0, 1]$, then with probability at least $1 - \delta$, the inequality $(\Delta_t - \hat{\Delta}_t)^2 \leq \theta_t^2$ holds simultaneously for all t .

Proof Sketch: First note that by Hölder's inequality,

$$(\Delta_t - \hat{\Delta}_t)^2 = ((\mathbf{w}_{t-1} - \mathbf{u})^\top \mathbf{x}_t)^2 \leq 2 \mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t d_{t-1}(\mathbf{w}_{t-1}, \mathbf{u}) . \quad (3)$$

The algorithm only performs an update when $Z_t = 1$. Since this update is that of online ridge regression, we can use techniques in (Azoury & Warmuth, 2001) to show that

$$\frac{1}{2} \sum_{i=1}^{t-1} Z_i \left((y_i - \hat{\Delta}'_i)^2 - (y_i - \Delta_i)^2 \right) \leq \frac{1}{2} - d_{t-1}(\mathbf{u}, \mathbf{w}_{t-1}) + 2 \sum_{i=1}^{t-1} Z_i r_i .$$

Plugging back into (3) gives

$$(\Delta_t - \hat{\Delta}_t)^2 \leq \mathbf{x}_t^T A_{t-1}^{-1} \mathbf{x}_t \left(1 + 4 \sum_{i=1}^{t-1} Z_i r_i - \sum_{i=1}^{t-1} Z_i \left((y_i - \hat{\Delta}'_i)^2 - (y_i - \Delta_i)^2 \right) \right) . \quad (4)$$

As in the proof of Lemma 3, we construct the martingale difference sequence $M_i = Z_i(\Delta_i - y_i)(\Delta_i - \hat{\Delta}'_i)$ and use tail bounds from (Kakade & Tewari, 2008) to prove that for any given $t > 1$, with probability at least $1 - \delta/t^2$,

$$-\frac{1}{2} \sum_{i=1}^{t-1} Z_i \left((y_i - \hat{\Delta}'_i)^2 - (y_i - \Delta_i)^2 \right) \leq 36 \log(t/\delta) .$$

Plugging the above into Eq. (4) and recalling the definition of θ_t , we have that $(\Delta_t - \hat{\Delta}_t)^2 \leq \theta_t^2$. A union bound over all t concludes the proof. \blacksquare

Remark 1 *Computing the intermediate vector \mathbf{w}'_{t-1} from \mathbf{w}_{t-1} , as defined in the Selective Sampler pseudocode, corresponds to projecting \mathbf{w}_{t-1} onto the convex set $C_t = \{\mathbf{w} \in \mathbb{R}^d : |\mathbf{w}^\top \mathbf{x}_t| \leq 1\}$ w.r.t. the Bregman divergence d_{t-1} , i.e., $\mathbf{w}'_{t-1} = \operatorname{argmin}_{\mathbf{u} \in C_t} d_{t-1}(\mathbf{u}, \mathbf{w}_{t-1})$. Notice that C_t includes the unit ball since \mathbf{x}_t is normalized. This projection step is needed for technical purposes during the construction of our bounded martingale difference sequence (see previous lemmas). Unlike similar constructions (e.g. (Hazan et al., 2006; Dani et al., 2008)), we do not project onto the unit ball, which would involve a line search over matrices and would slow down the algorithm to a significant extent. Moreover, we can prove that the total number of times that Selective Sampler projects onto C_t is $O(d^2 \log^2(T/\delta))$.*

2.4 An Online-to-Batch Conversion

It is instructive to see what the bound in Theorem 1 looks like when we assume that the instances \mathbf{x}_t are drawn i.i.d. according to an unknown distribution over the Euclidean unit sphere, and to compare this bound to standard statistical learning bounds. We model the distribution of the instances near the hyperplane $\{\mathbf{x} : \mathbf{u}^\top \mathbf{x} = 0\}$ using the well-known *Mammen-Tsybakov low noise condition*³ (Tsybakov, 2004):

$$\text{There exist } c > 0 \text{ and } \alpha \geq 0 \text{ such that } P(|\mathbf{u}^\top \mathbf{x}| < \epsilon) \leq c \epsilon^\alpha \text{ for all } \epsilon > 0.$$

We now describe a simple randomized algorithm which, with high probability over the sampling of the data, returns a linear predictor with a small expected risk (expectation is taken over the randomization of the algorithm). The algorithm is as follows:

1. Run Algorithm 1 with confidence level δ on the data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$, and obtain the sequence of predictors $\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_{T-1}$
2. Pick $r \in \{0, 1, \dots, T-1\}$ uniformly at random and return \mathbf{w}_r .

Due to the unavailability of all labels, standard conversion techniques that return a single deterministic hypothesis (e.g., (Cesa-Bianchi & Gentile, 2008)) do not readily apply here. The following theorem states a high probability bound on the risk and the label complexity of our algorithm. We omit the proof due to space constraints.

Theorem 7 *Let \mathbf{w}_r be the linear hypothesis returned by the above algorithm. Then with probability at least $1 - \delta$ we have*

$$\begin{aligned} \mathbb{E}_r [P'_r(\mathbf{y} \mathbf{w}_r^\top \mathbf{x} < 0)] &\leq P(\mathbf{y} \mathbf{u}^\top \mathbf{x} < 0) + \mathcal{O} \left((d \log(T/\delta))^{\frac{\alpha+1}{\alpha+2}} T^{-\frac{\alpha+1}{\alpha+2}} + \log \left(\frac{\log T}{\delta} \right) / T \right) , \\ N_T &= \mathcal{O} \left((d^2 \log^2(T/\delta))^{\frac{\alpha}{\alpha+2}} T^{\frac{2}{\alpha+2}} + \log(1/\delta) \right) , \end{aligned}$$

where \mathbb{E}_r is the expectation over the randomization in the algorithm, and $P'_r(\cdot)$ denotes the conditional probability⁴ $P(\cdot | \mathbf{x}_1, \dots, \mathbf{x}_{r-1}, y_1, \dots, y_{r-1})$.

³The constant c might actually depend on the input dimension d . For notational simplicity, Theorem 7 regards c as a constant, hence it is hidden in the big-oh notation.

⁴Notice the difference with the conditional probability $P_r(\cdot)$ defined in Section 2.1.

As α goes from 0 (no assumptions on the noise) to ∞ (hard separation assumption), the above bound on the average regret roughly interpolates between $1/\sqrt{T}$ and $1/T$. Correspondingly, the bound on the number of labels N_T goes from T to $\log^2 T$. In particular, observe that, viewed as a function of N_T (and disregarding log factors), the instantaneous regret is of the form $N_T^{-\frac{\alpha+1}{2}}$. These bounds are sharper than those in (Cavallanti et al., 2009) and, in fact, no further improvement is generally possible (see Castro and Nowak (2008)). The same rates are obtained by (Hanneke, 2009) under much more general conditions, for less efficient algorithms that are based on empirical risk minimization.

One might wonder whether an adaptively adversarial model of learning might somehow be overkill for obtaining i.i.d. results. As a matter of fact, the way our algorithm works makes an adaptively adversarial analysis a very natural one even for deriving the above i.i.d. results.

2.5 Related Work

Selective sampling is an online learning framework lying between passive learning (where the algorithm has no control over the learning sequence) and fully active learning (where the learning algorithm is allowed to select the instances \mathbf{x}_t). Recent papers on active learning include (Balcan et al., 2006; Bach, 2006; Balcan et al., 2007; Castro & Nowak, 2008; Dasgupta et al., 2008; Dasgupta et al., 2005; Hanneke, 2007; Hanneke, 2009). All of these papers consider the case when instances are drawn i.i.d. from a fixed distribution (either known or unknown). In particular, (Dasgupta et al., 2005) gives an efficient Perceptron-like algorithm for learning within accuracy ϵ the class of homogeneous d -dimensional half-spaces under the uniform distribution over the unit ball, with label complexity of the form $d \log \frac{1}{\epsilon}$. Still in the i.i.d. setting, more general results are given in (Balcan et al., 2007). A neat analysis of previously proposed general active learning schemes (Balcan et al., 2006; Dasgupta et al., 2008) is provided by the aforementioned paper (Hanneke, 2009). Due to their generality, many of the above results rely on schemes that are computationally prohibitive (exceptions being the results in (Dasgupta et al., 2005) and the realizable cases analyzed in (Balcan et al., 2007)). Finally, pool-based active learning scenarios are considered in (Bach, 2006, and the references therein), though the analysis is only asymptotic in nature and no quantification is given of the trade-off between risk and number of labels.

The results of Theorem 1 are more in line with the worst-case analyses in (Cesa-Bianchi et al., 2006; Strehl & Littman, 2008; Cesa-Bianchi et al., 2009). These papers present variants of Recursive Least Squares algorithms that operate on arbitrary instance sequences. The analysis in (Cesa-Bianchi et al., 2006) is completely worst case: the authors make no assumptions whatsoever on the mechanism generating instances or labels; however, they are unable to prove bounds on the label query rate. The setups in (Strehl & Littman, 2008; Cesa-Bianchi et al., 2009) are closest to ours in that they assume the same linear stochastic noise-model used in our analysis. The algorithm presented in (Strehl & Littman, 2008) approximates the Bayes margin to within a given accuracy ϵ , and queries $\tilde{O}(d^3/\epsilon^4)$ labels; this bound is significantly inferior to our bound, and it seems to hold only in the finite-dimensional case. A more precise comparison can be made to the (expectation) bounds presented in (Cesa-Bianchi et al., 2009), which are of the form $R_T \leq \min_{0 < \epsilon < 1} \left(\epsilon T_\epsilon + \frac{T^{1-\kappa}}{\epsilon^2} + \frac{d}{\epsilon^2} \ln T \right)$, and $N_T = \mathcal{O}(dT^\kappa \ln T)$, where $\kappa \in [0, 1]$ is a parameter of their algorithm. In contrast, our bound in Theorem 1 has a sharper dependence on ϵ , and a better trade-off between R_T and N_T . Moreover, unlike the analysis in (Cesa-Bianchi et al., 2009), our analysis covers the case where the instances are generated by an adaptive adversary.

3 The Multiple Teacher Case

The problem is still online binary classification, where at each time step $t = 1, 2, \dots$ the learner receives an input $\mathbf{x}_t \in \mathbb{R}^d$, with $\|\mathbf{x}_t\| \leq 1$, and outputs a binary prediction \hat{y}_t . However, there are now K available teachers, each with his own area of expertise. If \mathbf{x}_t falls within the expertise region of teacher j , then that teacher can provide an accurate label. After making each binary prediction, the learner chooses if to issue a query to one or more of the K teachers. The learner is free to query any subset of teachers, but each teacher charges a unit cost per label. The expertise region of each teacher is unknown to the learner, and can only be inferred indirectly from the binary labels purchased from that teacher.

Formally, we assume that teacher j is associated with a weight vector $\mathbf{u}_j \in \mathbb{R}^d$, where $\|\mathbf{u}_j\| \leq 1$. If teacher j is queried on round t , he stochastically generates the binary label $y_{j,t}$ according to the law $P_t(y_{j,t} = 1 | \mathbf{x}_t) = (1 + \Delta_{j,t})/2$, where $\Delta_{j,t} = \mathbf{u}_j^\top \mathbf{x}_t$ and, as in Section 2, \mathbf{x}_t can be chosen adversarially depending on previous \mathbf{x} 's and y_j 's. We consider $|\Delta_{j,t}|$ to be the *confidence* of teacher j in his label for \mathbf{x}_t . When the learner issues a query, he receives nothing other than the binary label itself, and the confidence is only part of our theoretical model of the teacher. If \mathbf{x}_t is almost orthogonal to \mathbf{u}_j then teacher j has a very low confidence in his label, and we say that \mathbf{x}_t lies outside the expertise region of teacher j .

It is no longer clear how we should evaluate the performance of the learner, since the K teachers will

often give inconsistent labels on the given \mathbf{x}_t , and we do not have a well defined ground truth to compare against. Intuitively, we would like the learner to predict the label of \mathbf{x}_t as accurately as the teachers who are experts on \mathbf{x}_t . To formalize this intuition, define the average margin of a generic subset of teachers⁵ $C \subseteq [K]$ as $\Delta_{C,t} = \frac{1}{|C|} \sum_{i \in C} \Delta_{i,t}$. We define the set of experts for each instance using a user-specified parameter $\tau > 0$. Define

$$j_t^* = \operatorname{argmax}_j |\Delta_{j,t}| \text{ and } C_t = \{i : |\Delta_{i,t}| \geq |\Delta_{j_t^*,t}| - \tau\} . \quad (5)$$

In words, j_t^* is the *most confident teacher* at time t , and C_t is the *set of confident teachers* at time t . This means that τ is a tolerance parameter that defines how confident a teacher must be, compared to the most confident teacher, to be considered a confident teacher. Although τ does not appear explicitly in the notation C_t , the reader should keep in mind that C_t and other sets defined later on in this section all depend on τ . Using the definitions above, $\Delta_{C_t,t}$ is the average margin of the confident teachers, and we abbreviate $\Delta_t = \Delta_{C_t,t}$.

Now, let y_t be the random variable that takes values in $\{-1, 1\}$, with $P_t(y_t = 1 | \mathbf{x}_t) = (1 + \Delta_t)/2$. In words, y_t is the binary label generated according to the average margin of the confident teachers. We consider the sequence y_1, \dots, y_T to be our ad-hoc ground-truth, and the goal of our algorithm is to accurately predict this sequence. Note that an equivalent way of generating y_t is by picking a confident teacher j uniformly at random from C_t and setting $y_t = y_{j,t}$. Indeed there are other reasonable ways to define the ground-truth for this problem, however, we feel that our definition coincides with our intuitions on learning from teachers with different areas of expertise. If τ is set to be 1, the learner is compared against the average margin of all K teachers, while if $\tau = 0$, the learner is compared against the single most confident teacher.

We now describe and analyze two algorithms within the multiple teacher setting. We call these algorithms “first version” and “second version”. In the first version, the algorithm queries either all of the teachers or none of the teachers. The second version is more refined in that the algorithm may query a different subset of teachers on each round.

3.1 Algorithm, First Version

The learner attempts to model each weight vector \mathbf{u}_j with a sequence of weight vectors $(\mathbf{w}_{j,t})_{t=1}^T$. As in the single teacher case, the learner maintains a variable threshold θ_t , which can be interpreted as the learner’s confidence in its current set of weight vectors. The learner attempts to mimic the process of generating y_t by choosing its own set of confident teachers at each time step. Denoting $\hat{\Delta}_{j,t} = \mathbf{w}_{j,t}^\top \mathbf{x}_t$, the learner defines

$$\hat{j}_t = \operatorname{argmax}_j |\hat{\Delta}_{j,t}| \text{ and } \hat{C}_t = \{i : |\hat{\Delta}_{i,t}| \geq |\hat{\Delta}_{\hat{j}_t,t}| - \tau - 2\theta_t\} ,$$

where \hat{j}_t is the learner’s estimate of the most confident teacher, and \hat{C}_t is the learner’s estimate of the set of confident teachers. Note that the definition of \hat{C}_t is more inclusive than the definition of C_t in Eq. (5), in that it also includes teachers whose confidence falls below $|\hat{\Delta}_{\hat{j}_t,t}| - \tau$. This accounts for the uncertainty regarding the learner’s set of weight vectors.

As above, we define the notation $\hat{\Delta}_{C,t} = \frac{1}{|C|} \sum_{i \in C} \hat{\Delta}_{i,t}$, and abbreviate $\hat{\Delta}_t = \hat{\Delta}_{\hat{C}_t,t}$. The learner predicts the binary label $\hat{y}_t = \operatorname{sgn}(\hat{\Delta}_t)$. Let P_t denote the conditional probability $P_t(\cdot) = \mathbb{P}(\cdot | \mathbf{x}_1, y_{1,1} \dots, y_{K,1}, \mathbf{x}_2, y_{1,2} \dots, y_{K,2}, \dots, \mathbf{x}_{t-1}, y_{1,t-1}, \dots, y_{K,t-1}, \mathbf{x}_t)$, and let the regret of the learner be

$$R_T = \sum_{t=1}^T \left(P_t(y_t \hat{\Delta}_t < 0) - P_t(y_t \Delta_t < 0) \right) . \quad (6)$$

Next, we proceed to describe our criterion for querying teachers. We present a simple criterion that either sets $Z_t = 1$ and queries all of the teachers or sets $Z_t = 0$ and queries none of them. Hence, the learner either incurs a cost of K or a cost of 0 on each round. We partition the set of confident teachers \hat{C}_t into two sets,

$$\begin{aligned} \hat{H}_t &= \{i : |\hat{\Delta}_{i,t}| \geq |\hat{\Delta}_{\hat{j}_t,t}| - \tau + 2\theta_t\} \\ \hat{B}_t &= \{i : |\hat{\Delta}_{\hat{j}_t,t}| - \tau - 2\theta_t \leq |\hat{\Delta}_{i,t}| < |\hat{\Delta}_{\hat{j}_t,t}| - \tau + 2\theta_t\} . \end{aligned}$$

\hat{H}_t is the set of teachers with especially high confidence, while \hat{B}_t is the set of teachers with borderline confidence. Intuitively, the learner is unsure whether the teachers in \hat{B}_t should or should not be included in \hat{C}_t . The learner issues a query (to all K teachers) if there exists a set $S \subseteq \hat{B}_t$ such that either $\hat{\Delta}_t \hat{\Delta}_{\hat{H}_t \cup S,t} < 0$ or $|\hat{\Delta}_{\hat{H}_t \cup S,t}| \leq \theta_t$. In other words, the learner searches for a subset of \hat{B}_t such that replacing \hat{B}_t with that subset would either flip the sign of $\hat{\Delta}_t$ or make it too small. If a query is issued, each weight vector $\mathbf{w}_{j,t}$ is updated as in the single teacher case. Pseudocode of this learner is given in Algorithm 2.

⁵Here and throughout, $[K] = \{1, 2, \dots, K\}$.

Algorithm 2: Multiple Teacher Selective Sampler – first version

input confidence level $\delta \in (0, 1]$, tolerance parameter $\tau \geq 0$
initialize $A_0 = I$, $\forall j \in [K]$ $\mathbf{w}_{j,0} = \mathbf{0}$
for $t = 1, 2, \dots$
 receive $\mathbf{x}_t \in \mathbb{R}^d$: $\|\mathbf{x}_t\| \leq 1$
 $\theta_t^2 = \mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t (1 + 4 \sum_{i=1}^{t-1} Z_i r_i + 36 \log(Kt/\delta))$
 $\forall j \in [K]$ $\hat{\Delta}_{j,t} = \mathbf{w}_{j,t-1}^\top \mathbf{x}_t$ and $\hat{j}_t = \operatorname{argmax}_j |\hat{\Delta}_{j,t}|$
 predict $\hat{y}_t = \operatorname{sgn}(\hat{\Delta}_t) \in \{-1, +1\}$
 $Z_t = \begin{cases} 1 & \text{if } \exists S \subseteq \hat{B}_t : \hat{\Delta}_t \hat{\Delta}_{S \cup \hat{H}_t, t} < 0 \text{ or } |\hat{\Delta}_{S \cup \hat{H}_t, t}| \leq \theta_t \\ 0 & \text{otherwise} \end{cases}$
 if $Z_t = 1$
 query $y_{1,t}, \dots, y_{K,t}$
 $A_t = A_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top$, $r_t = \mathbf{x}_t^\top A_t^{-1} \mathbf{x}_t$
 for $j = 1, \dots, K$
 $\mathbf{w}'_{j,t-1} = \begin{cases} \mathbf{w}_{j,t-1} - \left(\frac{|\hat{\Delta}_{j,t}| - 1}{\mathbf{x}_t^\top A_{t-1}^{-1} \mathbf{x}_t} \right) A_{t-1}^{-1} \mathbf{x}_t & \text{if } |\hat{\Delta}_{j,t}| > 1, \\ \mathbf{w}_{j,t-1} & \text{otherwise} \end{cases}$
 $\mathbf{w}_{j,t} = A_t^{-1} (A_{t-1} \mathbf{w}'_{j,t-1} + y_{j,t} \mathbf{x}_t)$
 else
 $A_t = A_{t-1}$, $r_t = 0$ and $\forall j \in [K]$ $\mathbf{w}_{j,t} = \mathbf{w}_{j,t-1}$

3.2 Analysis, First Version

Our learning algorithm relies on labels it receives from a set of teachers, and therefore our bounds should naturally depend on the ability of those teachers to provide accurate labels for the concrete sequence $\mathbf{x}_1, \dots, \mathbf{x}_T$. For example, if an input \mathbf{x}_t lies outside the expertise regions of all teachers, we cannot hope to learn anything from the labels provided by the teachers for this input. Similarly, there is nothing we can do on rounds where the set of confident teachers is split between two equally confident but conflicting opinions. We count these difficult rounds by defining, for any $\epsilon > 0$,

$$T_\epsilon = \sum_{t=1}^T \mathbb{I}\{|\Delta_t| \leq \epsilon\}. \quad (7)$$

The above is just a multiple teacher counterpart to (1). However it is interesting to note that even in a case where most teachers have low confidence in their prediction on any given round, T_ϵ can still be small provided that the experts in the field have a confident opinion.

A more subtle difficulty presents itself when the collective opinion expressed by the set of confident teachers changes qualitatively with a small perturbation of the input \mathbf{x}_t or one of the weight vectors \mathbf{u}_j . To state this formally, define for any $\epsilon > 0$

$$\begin{aligned} H_{\epsilon,t} &= \{i : |\Delta_{i,t}| \geq |\Delta_{j_t^*,t}| - \tau + \epsilon\} \\ B_{\epsilon,t} &= \{i : |\Delta_{j_t^*,t}| - \tau - \epsilon \leq |\Delta_{i,t}| < |\Delta_{j_t^*,t}| - \tau + \epsilon\}. \end{aligned}$$

The set $H_{\epsilon,t}$ is the subset of teachers in C_t with especially high confidence, ϵ higher than the minimal confidence required for inclusion in C_t . In contrast, the set $B_{\epsilon,t}$ is the set of teachers with borderline confidence: either teachers in C_t that would be excluded if their margin were smaller by ϵ , or teachers that are not in C_t that would be included if their margin were larger by ϵ . We say that the average margin of the confident teachers is *unstable* with respect to τ and ϵ if $|\Delta_t| > \epsilon$ but we can find a subset $S \subseteq B_{\epsilon,t}$ such that either $\Delta_t \Delta_{S \cup H_{\epsilon,t}, t} < 0$ or $|\Delta_{S \cup H_{\epsilon,t}, t}| < \epsilon$. In other words, we are dealing with the situation where Δ_t is sufficiently confident, but a small ϵ -perturbation to the margins of the individual teachers can cause its sign to flip,

or its confidence to fall below ϵ . We count the unstable rounds by defining, for any⁶ $\epsilon > 0$,

$$T'_\epsilon = \sum_{t=1}^T \mathbb{1}\{|\Delta_t| > \epsilon\} \mathbb{1}\{\exists S \subseteq B_{\epsilon,t} : \Delta_t \Delta_{S \cup H_{\epsilon,t},t} < 0 \vee |\Delta_{S \cup H_{\epsilon,t},t}| \leq \epsilon\}. \quad (8)$$

Intuitively T'_ϵ counts the number of rounds on which an ϵ -perturbation of the $\Delta_{t,j}$ of the teachers either changes the sign of the average margin or results in an average margin close to zero. Like T_ϵ , this quantity measures an inherent hardness of the multiple teacher problem.

The following theorem is the main theoretical result of this section. It provides an upper bound on the regret of the learner, as defined in Eq. (6), and on the total cost of queries, $N_T = K \sum_{t=1}^T Z_t$. Again, we stress both the data and the time-dependent aspects of the bound.

Theorem 8 *Assume Algorithm 2 is run with a confidence parameter $\delta > 0$. Then with probability at least $1 - \delta$ it holds for all $T > 0$ that*

$$\begin{aligned} R_T &\leq \inf_{\epsilon > 0} \left\{ \epsilon T_\epsilon + T'_\epsilon + \mathcal{O} \left(\frac{\log |A_T| \log(KT/\delta) + \log^2 |A_T|}{\epsilon^2} \right) \right\} \\ &= \inf_{\epsilon > 0} \left\{ \epsilon T_\epsilon + T'_\epsilon + \mathcal{O} \left(\frac{d^2 \log^2(KT/\delta)}{\epsilon^2} \right) \right\}, \\ N_T &\leq K \inf_{\epsilon > 0} \left\{ T_\epsilon + T'_\epsilon + \mathcal{O} \left(\frac{\log |A_T| \log(KT/\delta) + \log^2 |A_T|}{\epsilon^2} \right) \right\} \\ &= K \inf_{\epsilon > 0} \left\{ T_\epsilon + T'_\epsilon + \mathcal{O} \left(\frac{d^2 \log^2(KT/\delta)}{\epsilon^2} \right) \right\}. \end{aligned}$$

As in the proof of Theorem 1, we begin by decomposing the regret. For any $\epsilon > 0$, Lemma 9 states that $R_T \leq \epsilon T_\epsilon + T'_\epsilon + U_{T,\epsilon} + Q_{T,\epsilon}$, where T_ϵ is defined in Eq. (7), T'_ϵ is defined in Eq. (8), and

$$U_{T,\epsilon} = \sum_{t=1}^T \bar{Z}_t \mathbb{1}\{\Delta_t \hat{\Delta}_t < 0\}, \quad Q_{T,\epsilon} = \sum_{t=1}^T Z_t \mathbb{1}\{\forall S \subseteq B_{\epsilon,t} : \Delta_t \Delta_{S \cup H_{\epsilon,t}} \geq 0, |\Delta_{S \cup H_{\epsilon,t}}| > \epsilon\}.$$

T_ϵ and T'_ϵ deal with time steps on which the ground truth itself is unreliable, $U_{T,\epsilon}$ sums over rounds where the learner does not make a query, and $Q_{T,\epsilon}$ sums over rounds where a query is issued. Similarly, for any $\epsilon > 0$, Lemma 10 upper bounds the number of time steps on which a query is issued by $T_\epsilon + T'_\epsilon + Q_{T,\epsilon}$. Lemma 11 upper bounds $Q_{T,\epsilon}$ and Lemma 12 upper bounds $U_{T,\epsilon}$. Both lemmas rely on the assumption that $(\Delta_{j,t} - \hat{\Delta}_{j,t})^2 \leq \theta_t^2$ for all $t \in [T]$ and $j \in [K]$. A straightforward stratification of Lemma 6 in Section 2 over the K teachers verifies that this condition holds with high probability. The proofs of the mentioned lemmas are omitted.

Lemma 9 *For any $\epsilon > 0$ it holds that $R_T \leq \epsilon T_\epsilon + T'_\epsilon + U_{T,\epsilon} + Q_{T,\epsilon}$.*

Lemma 10 *For any $\epsilon > 0$, it holds that $\sum_{t=1}^T Z_t \leq T_\epsilon + T'_\epsilon + Q_{T,\epsilon}$.*

Lemma 11 *If $(\Delta_{j,t} - \hat{\Delta}_{j,t})^2 \leq \theta_t^2$ holds for all $j \in [K]$ and $t \in [T]$, then*

$$Q_{T,\epsilon} = \mathcal{O} \left(\frac{\log |A_T| \log(KT/\delta) + \log^2 |A_T|}{\epsilon^2} \right) = \mathcal{O} \left(\frac{d^2 \log^2(KT/\delta)}{\epsilon^2} \right).$$

Lemma 12 *If $(\Delta_{j,t} - \hat{\Delta}_{j,t})^2 \leq \theta_t^2$ for all $j \in [K]$ and $t \in [T]$, then $U_{T,\epsilon} = 0$ for all $\epsilon > 0$.*

3.3 Algorithm, Second Version

The second version differs from the first one in that now each teacher j has its own threshold $\theta_{j,t}$, and also its own matrix $A_{j,t}$. As a consequence, the set of confident teachers \hat{C}_t and the partition of \hat{C}_t into highly confident (\hat{H}_t) and borderline (\hat{B}_t) teachers have to be redefined as follows:

$$\begin{aligned} \hat{C}_t &= \{j : |\hat{\Delta}_{j,t}| \geq |\hat{\Delta}_{\hat{j}_t,t}| - \tau - \theta_{j,t} - \theta_{\hat{j}_t,t}\}, & \text{where } \hat{j}_t &= \operatorname{argmax}_j |\hat{\Delta}_{j,t}|, \\ \hat{H}_t &= \{i : |\hat{\Delta}_{i,t}| \geq |\hat{\Delta}_{\hat{j}_t,t}| - \tau + \theta_{j,t} + \max_{j \in \hat{C}_t} \theta_{j,t}\}, \\ \hat{B}_t &= \{i : |\hat{\Delta}_{\hat{j}_t,t}| - \tau - \theta_{j,t} - \theta_{\hat{j}_t,t} \leq |\hat{\Delta}_{i,t}| < |\hat{\Delta}_{\hat{j}_t,t}| - \tau + \theta_{j,t} + \max_{j \in \hat{C}_t} \theta_{j,t}\}. \end{aligned}$$

The pseudocode is given in Algorithm 3. Notice that the query condition defining Z_t now depends on an *average threshold* $\theta_{S \cup \hat{H}_t,t} = \frac{1}{|S \cup \hat{H}_t|} \sum_{j \in S \cup \hat{H}_t} \theta_{j,t}$.

⁶Notice that, up to degenerate cases, both T_ϵ and T'_ϵ tend to vanish as $\epsilon \rightarrow 0$. Hence, as in the single teacher case, the free parameter ϵ trades-off hardness terms against regret terms.

Algorithm 3: Multiple Teacher Selective Sampler – second version

input confidence level $\delta \in (0, 1]$, tolerance parameter $\tau \geq 0$
initialize $A_{j,0} = I$, $\mathbf{w}_{j,0} = \mathbf{0}$, $\forall j \in [K]$
for $t = 1, 2, \dots$
 receive $\mathbf{x}_t \in \mathbb{R}^d : \|\mathbf{x}_t\| \leq 1$
 $\forall j \in [K]$, $\theta_{j,t}^2 = \mathbf{x}_t^\top A_{j,t-1}^{-1} \mathbf{x}_t (1 + 4 \sum_{i=1}^{t-1} Z_i r_{j,i} + 36 \log(Kt/\delta))$
 $\forall j \in [K]$, $\hat{\Delta}_{j,t} = \mathbf{w}_{j,t-1}^\top \mathbf{x}_t$ and $\hat{j}_t = \operatorname{argmax}_j |\hat{\Delta}_{j,t}|$
 predict $\hat{y}_t = \operatorname{sgn}(\hat{\Delta}_{\hat{j}_t}) \in \{-1, +1\}$
 $Z_t = \begin{cases} 1 & \text{if } \exists S \subseteq \hat{B}_t : \hat{\Delta}_t \hat{\Delta}_{S \cup \hat{H}_t, t} < 0 \text{ or } |\hat{\Delta}_{S \cup \hat{H}_t, t}| \leq \theta_{S \cup \hat{H}_t, t} \\ 0 & \text{otherwise} \end{cases}$
 if $Z_t = 1$ and $j \in \hat{C}_t$
 query $y_{j,t}$
 $A_{j,t} = A_{j,t-1} + \mathbf{x}_t \mathbf{x}_t^\top$, $r_{j,t} = \mathbf{x}_t^\top A_{j,t}^{-1} \mathbf{x}_t$
 $\mathbf{w}'_{j,t-1} = \begin{cases} \mathbf{w}_{j,t-1} - \left(\frac{|\hat{\Delta}_{j,t}| - 1}{\mathbf{x}_t^\top A_{j,t-1}^{-1} \mathbf{x}_t} \right) A_{j,t-1}^{-1} \mathbf{x}_t & \text{if } |\hat{\Delta}_{j,t}| > 1, \\ \mathbf{w}_{j,t-1} & \text{otherwise} \end{cases}$
 $\mathbf{w}_{j,t} = A_{j,t}^{-1} (A_{j,t-1} \mathbf{w}'_{j,t-1} + y_{j,t} \mathbf{x}_t)$
 else
 $A_{j,t} = A_{j,t-1}$, $r_{j,t} = 0$ and $\mathbf{w}_{j,t} = \mathbf{w}_{j,t-1}$

3.4 Analysis, Second Version

The following theorem bounds the cumulative regret and the total number of queries with high probability. The proof is similar to the proof of Theorem 8. We keep the definitions of the sets $H_{\epsilon,t}$ and $B_{\epsilon,t}$ as given in Section 3.2, but in the bound on N_T in Theorem 13, we replace T'_ϵ with the more refined quantity T''_ϵ , where

$$T''_\epsilon = \sum_{t=1}^T \frac{|H_{\epsilon,t} \cup B_{\epsilon,t}|}{K} \mathbb{1}\{|\Delta_t| > \epsilon\} \mathbb{1}\{\exists S \subseteq B_{\epsilon,t} : \Delta_t \Delta_{S \cup H_{\epsilon,t}, t} < 0 \vee |\Delta_{S \cup H_{\epsilon,t}, t}| \leq \epsilon\}.$$

Note that T''_ϵ is similar to T'_ϵ except that while T'_ϵ only counts the number of times that perturbations to the $\Delta_{j,t}$'s lead to conflict or low confidence predictions, T''_ϵ counts the fraction of confident teachers involved in the conflict. If for most \mathbf{x}_t only a few of the K teachers are experts (highly confident), then one would expect T''_ϵ to be much smaller than T'_ϵ and thus we expect the number of queries to be small.

Theorem 13 *Assume Algorithm 3 is run with a confidence parameter $\delta > 0$. Then with probability at least $1 - \delta$ it holds for all $T > 0$ that*

$$\begin{aligned} R_T &\leq \inf_{\epsilon > 0} \left\{ \epsilon T_\epsilon + T'_\epsilon + \mathcal{O} \left(\frac{K \log |A_T| \log(KT/\delta) + K \log^2 |A_T|}{\epsilon^2} \right) \right\} \\ &= \inf_{\epsilon > 0} \left\{ \epsilon T_\epsilon + T'_\epsilon + \mathcal{O} \left(\frac{K d^2 \log^2(KT/\delta)}{\epsilon^2} \right) \right\}, \\ N_T &\leq K \inf_{\epsilon > 0} \left\{ T_\epsilon + T''_\epsilon + \mathcal{O} \left(\frac{K \log |A_T| \log(KT/\delta) + K \log^2 |A_T|}{\epsilon^2} \right) \right\} \\ &= K \inf_{\epsilon > 0} \left\{ T_\epsilon + T''_\epsilon + \mathcal{O} \left(\frac{K d^2 \log^2(KT/\delta)}{\epsilon^2} \right) \right\}. \end{aligned}$$

Note that the above theorem holds at the cost of losing a factor K elsewhere in the regret terms, thereby making Theorem 8 and Theorem 13 incomparable.

4 Conclusions and Ongoing Research

We introduced a new Ridge-Regression-like algorithm operating in a robust selecting sampling environment, where the adversary can adapt on the fly to the algorithm's choices. We gave sharp bounds on the cumulative

regret and the number of queries made by this algorithm, solving questions left open in previous investigations. We then lifted this machinery to solving the more involved problem where multiple unreliable teachers are available. We gave two algorithms and corresponding analyses.

We are currently running experiments on real-world data (the experimental setting is somewhat similar to the one described in (Donmez & Carbonell, 2008)) to see the performance of the multiple teacher algorithms compared to the simple baseline where K independent instances of the single teacher algorithm (Algorithm 1) are run in parallel. An implementation issue of the multiple teacher algorithms we have presented is the exponential explosion that seemingly arises when computing Z_t , due to the need to check all possible subsets $S \subseteq \hat{B}_t$. As a matter of fact, this check can be computed efficiently by sorting the teachers according to their estimated confidence $|\hat{\Delta}_{j,t}|$. Though preliminary, our experiments suggest that the multiple teacher algorithm largely outperforms the baseline, both in terms of accuracy and total number of requested labels.

On the theoretical side, a few points we are presently investigating are the following: i) The bound on N_T in Theorem 1 is tight w.r.t. ϵ (see the lower bound in (Cesa-Bianchi et al., 2009)), but need not be tight w.r.t. d . This might be due to the way we constructed our martingale argument to prove Lemma 6. ii) As a more general issue, we are trying to generalize our results to further label noise models, such as logistic models. iii) The bounds for the multiple teacher algorithms in Theorems 8 and 13 are likely to be suboptimal, and we are currently trying to better exploit the interaction structure among teachers. iv) Proactive learning, as presented in (Donmez & Carbonell, 2008; Yang & Carbonell, 2009b; Yang & Carbonell, 2009a), also allows for different costs for different teachers, the idea being that more expensive teachers may be more reliable. We are trying to see whether we can incorporate costs into our multiple teacher analysis.

Acknowledgments We thank the COLT 2010 reviewers for their helpful comments. This research was done while the second and the third authors were visiting Microsoft Research at Redmond. The second author acknowledges the PASCAL2 Network of Excellence under EC grant 216886 for supporting travel expenses to the conference.

References

- Azoury, K. S., & Warmuth, M. K. (2001). Relative loss bounds for online density estimation with the exponential family of distributions. *Machine Learning*, 43, 211–246.
- Bach, F. (2006). *Active learning for misspecified generalized linear models* (Technical Report N15/06/MM). Ecole des mines de Paris.
- Balcan, M., Beygelzimer, M., & Langford, J. (2006). Agnostic active learning. *Proc. of the 23th International Conference on Machine Learning* (pp. 65–72).
- Balcan, M., Broder, A., & T. Zhang, T. (2007). Margin-based active learning. *Proc. of the 20th Annual Conference on Learning Theory* (pp. 35–50).
- Castro, R., & Nowak, R. D. (2008). Minimax bounds for active learning. *IEEE Trans. IT*, 54, 2339–2353.
- Cavallanti, G., Cesa-Bianchi, N., & Gentile, C. (2009). Linear classification and selective sampling under low noise conditions. *Advances in Neural Information Processing Systems 21*.
- Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2003). Learning probabilistic linear-threshold classifiers via selective sampling. *Proc. of the 16th Annual Conference on Learning Theory* (pp. 373–387).
- Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2005). A second-order Perceptron algorithm. *SIAM Journal on Computing*, 43, 640–668.
- Cesa-Bianchi, N., & Gentile, C. (2008). Improved risk tail bounds for on-line algorithms. *IEEE Trans. IT*, 54, 386–390.
- Cesa-Bianchi, N., Gentile, C., & Orabona, F. (2009). Robust bounds for classification via selective sampling. *Proc. of the 26th International Conference on Machine Learning*.
- Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Worst-case analysis of selective sampling for linear classification. *JMLR*, 7, 1025–1230.
- Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press.
- Cohn, R., Atlas, L., & Ladner, R. (1990). Training connectionist networks with queries and selective sampling. *Advances in Neural Information Processing Systems 2*.

- Dani, V., Hayes, T. P., & Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback. *Proc. of the 12th Annual Conference on Learning Theory*.
- Dasgupta, S., Hsu, D., & Monteleoni, C. (2008). A general agnostic active learning algorithm. *Advances in Neural Information Processing Systems 21*.
- Dasgupta, S., Kalai, A. T., & Monteleoni, C. (2005). Analysis of perceptron-based active learning. *Proc. of the 18th Annual Conference on Learning Theory*.
- Dekel, O., Gentile, C., & Sridharan, K. (2010). *Robust selective sampling from single and multiple teachers* (Technical Report). Microsoft Research, Università dell'Insubria, TTI.
- Donmez, P., & Carbonell, J. G. (2008). Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. *CIKM*.
- Freund, Y., Seung, S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, 28, 133–168.
- Hanneke, S. (2007). A bound on the label complexity of agnostic active learning. *Proc. of the 24th International Conference on Machine Learning* (pp. 353–360).
- Hanneke, S. (2009). Adaptive rates of convergence in active learning. *Proc. of the 22th Annual Conference on Learning Theory*.
- Hazan, E., Kalai, A., Kale, S., & Agarwal, A. (2006). Logarithmic regret algorithms for online convex optimization. *Proc. of the 19th Annual Conference on Learning Theory*.
- Hoerl, A., & Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12, 55–67.
- Kakade, S., & Tewari, A. (2008). On the generalization ability of online strongly convex programming algorithms. *Advances in Neural Information Processing Systems*.
- Lai, T. L., & Wei, C. Z. (1982). Least squares estimates in stochastic regression models with applications to identification and control of dynamic systems. *The Annals of Statistics*, 154–166.
- Li, L., Littman, M., & Walsh, T. (2008). Knows what it knows: a framework for self-aware learning. *Proc. of the 25th International Conference on Machine Learning* (pp. 568–575).
- Strehl, A., & Littman, M. (2008). Online linear regression and its application to model-based reinforcement learning. *Advances in Neural Information Processing Systems 20*.
- Tsybakov, A. (2004). Optimal aggregation of classifiers in statistical learning. *Ann. of Stat.*, 32, 135–166.
- Vovk, V. (2001). Competitive on-line statistics. *International Statistical Review*, 69, 213–248.
- Yang, L., & Carbonell, J. (2009a). *Adaptive proactive learning with cost-reliability tradeoff* (Technical Report CMU-ML-09-114). Carnegie Mellon University.
- Yang, L., & Carbonell, J. (2009b). *Cost complexity of proactive learning via a reduction to realizable active learning* (Technical Report CMU-ML-09-113). Carnegie Mellon University.

Improved Guarantees for Agnostic Learning of Disjunctions

Pranjal Awasthi

Carnegie Mellon University
pawasthi@cs.cmu.edu

Avrim Blum

Carnegie Mellon University
avrim@cs.cmu.edu

Or Sheffet

Carnegie Mellon University
osheffet@cs.cmu.edu

Abstract

Given some arbitrary distribution \mathcal{D} over $\{0, 1\}^n$ and arbitrary target function c^* , the problem of agnostic learning of disjunctions is to achieve an error rate comparable to the error OPT_{disj} of the best disjunction with respect to (\mathcal{D}, c^*) . Achieving error $O(n \cdot \text{OPT}_{disj}) + \epsilon$ is trivial, and Winnow [13] achieves error $O(r \cdot \text{OPT}_{disj}) + \epsilon$, where r is the number of relevant variables in the best disjunction. In recent work, Peleg [14] shows how to achieve a bound of $\tilde{O}(\sqrt{n} \cdot \text{OPT}_{disj}) + \epsilon$ in polynomial time. In this paper we improve on Peleg’s bound, giving a polynomial-time algorithm achieving a bound of

$$O(n^{1/3+\alpha} \cdot \text{OPT}_{disj}) + \epsilon$$

for any constant $\alpha > 0$. The heart of the algorithm is a method for weak-learning when $\text{OPT}_{disj} = O(1/n^{1/3+\alpha})$, which can then be fed into existing agnostic boosting procedures to achieve the desired guarantee.

1 Introduction

Learning disjunctions (or conjunctions) over $\{0, 1\}^n$ in the PAC model is a well-studied and easy problem. The simple “list-and-cross-off” algorithm runs in linear time per example and requires only $O(n/\epsilon)$ examples to achieve error ϵ (ignoring the logarithmic dependence on the confidence term δ). The similarly efficient Winnow algorithm [13] requires only $O((r \log n)/\epsilon)$ examples to learn well when the target function is a disjunction of size r .

However, when the data is only “mostly” consistent with a disjunction, the problem becomes substantially harder. In this *agnostic* setting, our goal is to produce a hypothesis h whose error rate $\text{err}_{\mathcal{D}}(h) = \Pr_{\mathcal{D}}(h(x) \neq c^*(x))$ satisfies $\text{err}_{\mathcal{D}}(h) \leq c \cdot \text{OPT}_{disj} + \epsilon$, where OPT_{disj} is the error rate of the *best* disjunction and c is as small as possible. For example, while Winnow performs well as a function of the number of *attribute errors* of the best disjunction¹ [12, 1], this can be a factor $O(r)$ worse than the number of *mistakes* of the best disjunction. Recently, Feldman [3] has shown that for any constant $\epsilon > 0$, determining whether the best disjunction for a given dataset S has error $\leq \epsilon$ or error $\geq \frac{1}{2} - \epsilon$ is NP-hard. Even more recently, Feldman et al. [5] extend this hardness result to the problem of agnostic learning disjunctions by the hypothesis class of halfspaces. Thus, these results show that the problem of finding a disjunction (or linear separator) of error at most $\frac{1}{2} - \epsilon$ given that the error OPT_{disj} of the best disjunction is at most ϵ is computationally hard for any constant $\epsilon > 0$.

Given these hardness results, it is natural to consider what kinds of learning guarantees *can* be achieved. If the error OPT_{disj} of the best disjunction is $O(1/n)$ then learning is essentially equivalent to the noise-free case. Peleg [14] shows how to improve this to a bound of $\tilde{O}(1/\sqrt{n})$. In particular, on any given dataset S , his algorithm produces a disjunction of error rate on S at most $\tilde{O}(\sqrt{n} \cdot \text{OPT}_{disj}(S))$.²

¹The minimum number of variables that would need to be flipped in order to make the data perfectly consistent with a disjunction. This is essentially the same as its hinge loss.

²His results are for the “Red-Blue Set-Cover Problem” [2] which is equivalent to the problem of approximating the best disjunction, except that positive examples must be classified correctly (i.e., the goal is to approximate the minimum number of mistakes on negatives subject to correctly classifying the positives). The extension to allowing for two-sided error, however, is immediate.

In this work, we improve on the result of Peleg [14], achieving a bound of $O(n^{1/3+\alpha} \cdot \text{OPT}_{\text{disj}}) + \epsilon$ for any constant $\alpha > 0$, though our algorithm is not a “proper” learner (does not produce a disjunction as its output).³ In particular, our main result is an algorithm for weak-learning under an arbitrary distribution D , under the assumption that the optimal disjunction has error rate $O(1/n^{1/3+\alpha})$, which we can then feed into boosting procedures of [7, 9] or the recent ABOOSTDI booster of [4], to achieve the claimed guarantee.

Note that our guarantee holds for any distribution over $\{0, 1\}^n$. In contrast, most recent work on agnostic learning has been for the case of uniform or other “nice” distributions [8, 10, 11].

1.1 Our Results

We present a learning algorithm whose error rate is an $O(n^{1/3+\alpha})$ approximation to that of the best disjunction, for any $\alpha > 0$. Formally, we prove the following theorem.

Theorem 1 *There exists an algorithm that for an arbitrary distribution \mathcal{D} over $\{0, 1\}^n$ and arbitrary target function $c^* : \{0, 1\}^n \mapsto \{1, -1\}$, for every constant $\alpha > 0$ and every $\epsilon, \delta > 0$, runs in time polynomial in $1/\epsilon$, $\log(1/\delta)$, and n , uses $\text{poly}(1/\epsilon, \log(1/\delta), n)$ random examples from \mathcal{D} , and outputs a hypothesis h , such that with probability $> 1 - \delta$,*

$$\mathbf{err}_{\mathcal{D}}(h) \leq O(n^{\frac{1}{3}+\alpha} \text{OPT}_{\text{disj}}) + \epsilon$$

where $\text{OPT}_{\text{disj}} = \min_{f \in \text{DISJUNCTIONS}} \mathbf{err}_{\mathcal{D}}(f)$.

The proof of Theorem 1 is based on finding a weak-learner under the assumption that $\text{OPT} \equiv \text{OPT}_{\text{disj}} = O(n^{-(1/3+\alpha)})$. In particular, we show:

Theorem 2 *There exists an algorithm with the following property. For every distribution \mathcal{D} over $\{0, 1\}^n$ and every target function c^* such that $\text{OPT} < n^{-\frac{1}{3}-\alpha}$, for some constant $\alpha > 0$, for every $\delta > 0$, the algorithm runs in time $t(\delta, n)$, uses $m(\delta, n)$ random samples drawn from \mathcal{D} and outputs a hypothesis h , such that with probability $> 1 - \delta$,*

$$\mathbf{err}_{\mathcal{D}}(h) \leq \frac{1}{2} - \gamma$$

where t and m are polynomials in $n, 1/\delta$, and $\gamma = \Omega(n^{-2})$.

The high-level idea of the algorithm and proof for Theorem 2 is as follows. First, we can assume the target function is balanced (nearly equal probability mass on positive and negative examples) and that similarly no individual variable is noticeably correlated with the target, else weak-learning is immediate. So, for each variable i , the probability mass of positive examples with $x_i = 1$ is approximately equal to the fraction of negative examples with $x_i = 1$. Let c^{opt} denote the (unknown) optimal disjunction, which we may assume is monotone by including negated variables as additional features. Let r denote the number of relevant variables; i.e., the number of variables in c^{opt} . Also, assume for this discussion that we know the value of $\text{OPT} = \mathbf{err}_{\mathcal{D}}(c^{\text{opt}})$. Call an example x “good” if $c^*(x) = c^{\text{opt}}(x)$ and “bad” otherwise. Now, since the only negative examples that can have a relevant x_i set to 1 are the bad negatives, this means that for relevant variables i , $\Pr_{x \sim \mathcal{D}}(x_i = 1 | c^*(x) = -1) = O(\text{OPT})$. Therefore, $\Pr_{x \sim \mathcal{D}}(x_i = 1 | c^*(x) = +1) = O(\text{OPT})$ and so $\Pr_{x \sim \mathcal{D}}(x_i = 1) = O(\text{OPT})$ as well. This means that by estimating $\Pr_{x \sim \mathcal{D}}(x_i = 1)$ for each variable i , we can remove all variables of density $\omega(\text{OPT})$ from the system, knowing they are irrelevant.

At this point, we have nearly all the ingredients for the $\tilde{O}(1/\sqrt{n})$ bound of Peleg [14]. In particular, since all variables have density $O(\text{OPT})$, this means the average number of variables set to 1 per example is $O(\text{OPT} \cdot n)$. Let S' be the set of examples whose density is at most twice the average (so $\Pr(S') \geq 1/2$); we now claim that if $\text{OPT} = o(1/\sqrt{n})$, then either S' is unbalanced or else some variable x_i must have noticeable correlation with the target over examples in S' . In particular, since positive examples must have on average at least $1 - O(\text{OPT})$ relevant variables set to 1, and the good negative examples have zero relevant variables set to 1, the only way for S' to be balanced and have no relevant variable with noticeable correlation is for the bad negative examples to on average have $\Omega(1/\text{OPT})$ relevant variables set to 1. But this is not possible since all examples in S' have only $O(\text{OPT} \cdot n)$ variables set to 1, and $1/\text{OPT} \gg \text{OPT} \cdot n$ for $\text{OPT} = o(1/\sqrt{n})$. So, some hypothesis of the form: “if $x \notin S'$ then flip a fair coin, else predict x_i ” must be a weak-learner.

³This bound hides a low-order term of $(\log n)^{1/\alpha}$. Solving for equality yields $\alpha = \sqrt{\frac{\log \log n}{\log n}}$ and a bound of $O(n^{1/3+o(1)})$.

In order to improve over the $\tilde{O}(1/\sqrt{n})$ bound of [14], we do the following. Assume all variables have nearly the same density and all examples have nearly the same density as well. This is *not* without loss of generality (and the general case adds additional complications that must be addressed), but simplifies the picture for this high-level sketch. Now, if no individual variable or its complement is a weak predictor, by the above analysis it must be the case that the bad negative examples on average have a substantial number of variables set to 1 in the relevant region (essentially so that the total hinge-loss (attribute-errors) is $\Omega(m)$). Suppose now that one “guesses” such a bad negative example e and focuses on only those n' variables set to 1 by e . The disjunction c^{opt} restricted to this set may now make many mistakes on positive examples (the “substantial number of variables set to 1 in the relevant region” in e may still be a small fraction of the relevant region). On the other hand, because we have restricted to a relatively small number of variables n' , the *average density* of examples as a function of n' has dropped significantly.⁴ As a result, suppose we again discard all examples with a number of 1’s in these n' variables substantially larger than the average. Then, on the remainder, the *hinge-loss* (attribute-errors) caused by the bad negative examples is now substantially reduced. This more than makes up for the additional error on positive examples. In particular, we show one can argue that for *some* bad negative example e , if one performs the above procedure, then with respect to the remaining subset of examples, some variable must be a weak predictor. In the end, the final hypothesis is defined by an example e , a threshold θ , and a variable i , and will be of the form “if $x \cdot e \notin [1, \theta]$ then flip a coin, else predict x_i .” The algorithm then simply searches over all such triples. In the general case (when the variables and the examples do not all have the same density), this is preceded by a preprocessing step that groups variables and examples into a number of buckets and then runs the above algorithm on each bucket.

The rest of the paper is organized as follows. We start off with notation and definitions in Section 2. In Section 3 we prove Theorem 1. We achieve this in two steps: first we show how to get a weak learner for the special case that the examples and variables are fairly homogeneous (all variables set to 1 roughly the same number of times, and all examples with roughly the same number of variables set to 1 (actually a somewhat weaker condition than this)). We then show how to reduce a general instance to this special case. In Section 3.3 we use existing boosting algorithms combined with this weak-learner to prove our main result. Finally, we discuss conclusions and future directions in Section 4.

2 Notation and Preliminaries

Let $\mathcal{X} = \{0, 1\}^n$ and let \mathcal{D} be the data distribution over \mathcal{X} . We have a labeling function $c^* : \mathcal{X} \rightarrow \{1, -1\}$, and use c^{opt} to denote the disjunction of least error with respect to c^* . Without loss of generality we may assume c^{opt} is monotone, and we denote the error rate of c^{opt} as OPT_{disj} or simply OPT . I.e., $\text{OPT} = \Pr_{x \sim \mathcal{D}}[c^{opt}(x) \neq c^*(x)]$. For the rest of the paper, we will assume that $\text{OPT} = \Omega(\frac{1}{\sqrt{n}})$ (otherwise we can use Peleg’s algorithm described in the previous section). We will also assume that we know the value of OPT .⁵ We use r to denote the number of variables in c^{opt} and we call these the *relevant* variables. We will call the examples on which c^* and c^{opt} agree “good”, and those on which c^* and c^{opt} disagree “bad”. The examples causing the most difficulty will be the bad negative examples, which can potentially satisfy many relevant variables, thus incurring up to r attribute errors (hinge-loss) and yet be labeled negative.

We assume that the algorithm gets as input $2m$ examples out of which m^+ are positive examples (their label is 1), and m^- are negative (their label is -1). We can assume for the goal of weak-learning that $m^+, m^- = m(1 \pm o(1))$, else we have an immediate weak predictor. Let m_{bad}^+ denote the number of bad positive examples, i.e., positives that do not satisfy c^{opt} , and let m_{bad}^- denote the number of bad negative examples, i.e., negatives that do satisfy c^{opt} . For convenience of notation (losing at most a factor of 2 in our guarantee) we assume that the error rate of c^{opt} on both positive and negative examples separately is at most OPT . Given this, we may assume that $m_{bad}^+ \leq m \cdot \text{OPT}(1 + o(1))$ and $m_{bad}^- \leq m \cdot \text{OPT}(1 + o(1))$.

Our algorithm will examine a set of $\tilde{O}(mn^2)$ hypotheses, of which we will prove that at least one has training error at most $1/2 - \tilde{\Omega}(1/n^2)$, under the assumption that OPT is $O(1/n^{1/3+\alpha})$. In the following we assume that m is sufficiently large, $\tilde{O}(n^4)$, so that with high probability this implies error at most $1/2 - \tilde{\Omega}(1/n^2)$ over \mathcal{D} . In particular, each hypothesis is defined by a training example, a threshold and a variable, and so by compression bounds [6], $\tilde{O}(n^4)$ training examples are sufficient to produce a weak learner with

⁴E.g., given two *random* vectors with $n' = n^{2/3}$ 1’s, their intersection would have expected size $(n')^{1/2}$. Of course, our dataset need not be uniform random examples of the given density, but the fact that all variables have the same density allows one to make a similar argument.

⁵If OPT is unknown, we can efficiently enumerate over possible guesses for OPT such that one such guess will be within a $1/poly$ additive factor of the true value. For each guess, we can run our algorithm and test it on a fresh sample to see if its output is a weak learner.

high probability.

Finally, it will be convenient to think of algorithms that make predictions on only a subset of the domain. If an algorithm predicts on a subset of probability mass p , and has error rate $1/2 - \gamma'$ on that subset, then by flipping a fair coin on the remainder, the overall error rate will be $1/2 - \gamma$ for $\gamma = p\gamma'$.

3 Proof of Theorem 1

We first build a weak agnostic learner for the best disjunction problem. Our weak learner has the guarantee given in Theorem 2, which we restate below.

Theorem 2 *There exists an algorithm with the following property. For every distribution \mathcal{D} over $\{0, 1\}^n$ and every target function c^* such that $\text{OPT} < n^{-\frac{1}{3}-\alpha}$, for some constant $\alpha > 0$, for every $\delta > 0$, the algorithm runs in time $t(\delta, n)$, uses $m(\delta, n)$ random samples drawn from \mathcal{D} and outputs a hypothesis h , such that with probability $> 1 - \delta$,*

$$\text{err}_{\mathcal{D}}(h) \leq \frac{1}{2} - \gamma$$

where t and m are polynomials in n , $1/\delta$, and $\gamma = \Omega(n^{-2})$.

Our algorithm has two stages: a preprocessing step (which we present later in Section 3.2) that ensures that all variables are set to 1 roughly the same number of times and that the bad and good examples have roughly the same number 1s, and a core algorithm (which we present first in Section 3.1) that operates on data of this form. One aspect of the preprocessing step is that in addition to partitioning examples into buckets, it may involve discarding some relevant variables, yielding a dataset in which only some $\tilde{m} \geq m/\text{polylog}(n)$ positive examples satisfy c^{opt} over the variables remaining. Thus, our assumption in Section 3.1 is that while the dataset has the “homogeneity” properties desired and the fraction of bad negative examples is $\text{OPT}(1 + o(1))$, the fraction of bad *positive* examples may be as large as $1 - 1/\text{polylog}(n)$. Nonetheless, this will still allow for weak learning.

3.1 (B, α, \tilde{m}) -Sparse Instances

As mentioned above, in this section we give a weak learning algorithm for a dataset that has certain “nice” homogeneity properties. We call such a dataset a (B, α, \tilde{m}) -sparse instance. We begin by describing what these properties are.

The first property is that there exists a positive integer B such that for each variable x_i , the number of positive examples in the instance with $x_i = 1$ is between $B/2$ and B , and the number of negative examples with $x_i = 1$ is between $\frac{B}{2}(1 - o(1))$ and $B(1 + o(1))$.

The first property implies that in this case the overall number of 1s in all examples is at most $2nB(1 + o(1))$, and therefore, an average example has no more than $\frac{nB(1+o(1))}{m}$ variables set to 1. If the bad negatives were typical examples, we would expect them to contain at most $\frac{nB}{m} \cdot m_{\text{bad}}(1 + o(1)) \leq nB \cdot \text{OPT}(1 + o(1))$ ones in total. While in general this may not necessarily be the case, we assume for this section that at least they are not *too* atypical on average. In particular, the second property we assume this instance satisfies is that the overall number of ones present in all the bad negatives is at most $n^{1+\alpha} B \text{OPT}$.

Denote by \tilde{m} the number of positive examples that c^{opt} classifies correctly. The third property is that $\tilde{m} \geq m/n^{o(\alpha)}$. If this dataset were our given training set then this would be redundant, as we already assume the stronger condition that the fraction of good positive examples is $1 - O(\text{OPT})$.⁶ However, \tilde{m} will be of use in later sections, when we call this algorithm as a subroutine on instances defined by only a subset of all the variables. In other words, we show here that even if we allow c^{opt} to make more mistakes on the positive examples (and in particular, to label almost all positives incorrectly!) yet make at most $m\text{OPT}$ mistakes on the negatives, we are still able to weak-learn. As our analysis shows, the condition we require of \tilde{m} is that the ratio $\frac{\tilde{m}}{m}$ dominates the ratio $\frac{\text{OPT}}{n^{-1/3}}$. Furthermore, the ratio $\frac{\tilde{m}}{m}$ will play a role in the definition of γ , our advantage over a random guess.

An instance satisfying all the above three properties is called a (B, α, \tilde{m}) -sparse instance. Next, we show how to get a weak learner for such sparse instances. We first introduce the following definitions.

Definition 3 *Given an example e and a positive integer threshold θ , we define the (e, θ) -restricted domain to be the set of all examples whose intersection with e is strictly smaller than θ . That is, the set of examples x such that $x \cdot e < \theta$. For any hypothesis h , we define the (e, θ) -restricted hypothesis to be h over any example*

⁶Indeed, if the original instance was sparse, we would have $\tilde{m} = m(1 - o(1))$.

that belongs to the (e, θ) restricted domain, and “I don’t know” (flipping a fair coin) over any other example. In particular, we consider the

- (e, θ) -restricted $(+1)$ -hypothesis – predict $+1$ if the given example intersects e on less than θ variables.
- (e, θ) -restricted (-1) -hypothesis – predict -1 if the given example intersects e on less than θ variables.
- (e, θ) -restricted x_i -hypothesis – predict $+1$ if the given example intersects e on less than θ variables and has $x_i = 1$.

We call these $n + 2$ restricted hypotheses the (e, θ) -restricted base hypotheses.

Our weak-learning algorithm enumerates over all pairs of (e, θ) , where e is a negative example in our training set and θ is an integer between 1 and n . For every such pair, our algorithm checks whether any of the $n + 2$ restricted hypothesis is a $\Omega(\frac{\tilde{m}}{m} \cdot \frac{\text{OPT}}{r})$ -weak-learner (see Algorithm 1 below). Our next lemma proves that for (B, α, \tilde{m}) -sparse instances, this algorithm indeed finds a weak-learner. In fact, we show that for every negative example e , it suffices to consider a particular value of θ .

Algorithm 1 A weak learner for sparse instances.

Input: A (B, α, \tilde{m}) sparse instance.

Step 1: For every negative example e in the set and every $\theta \in \{1, 2, \dots, n\}$

Step 1a: Check if any of the (e, θ) -restricted hypotheses from Definition 3 is a weak learner with error at most $\frac{1}{2} - \Omega(n^{-2})$.

Step 1b: If Yes, then output the corresponding hypothesis and halt.

Step 2: If no restricted hypothesis is a weak learner, output failure.

Lemma 4 Suppose we are given a (B, α, \tilde{m}) -sparse instance, and that c^{opt} makes no more than a $n^{-(\frac{1}{3} + \alpha)}$ fraction of errors on the negative examples. Then there exists a bad negative example e and a threshold θ such that one of the (e, θ) -restricted base hypotheses mentioned in Definition 3 has error at most $1/2 - \gamma$ for $\gamma = \Omega(\frac{\tilde{m}}{m} \cdot \frac{\text{OPT}}{r})$. Since we may assume $\text{OPT} > 1/\sqrt{n}$, this implies $\gamma = \Omega(n^{-2})$. Thus Algorithm 1 outputs a hypothesis of error at most $\frac{1}{2} - \Omega(n^{-2})$.

Proof: Let m^+ and m^- be the number of positive and negative examples in this sparse instance, where we reserve m to refer to the size of the original dataset of which this sparse instance is a subset. As before, call examples “good” if they are classified correctly by c^{opt} , else call them “bad”. We know $B = O(m\text{OPT})$, because relevant variables have no more than $O(m\text{OPT})$ occurrences of 1 over the negative examples. Since each good positive example has to have at least one relevant variable set to 1, it must also hold that $B = \Omega(\tilde{m}/r)$. It follows that $r\text{OPT} = \Omega(\tilde{m}/m)$. We now show how to find a weak learner given a (B, α, \tilde{m}) -sparse instance, based on a bad negative example.

Consider any bad negative example e_i with t_i variables set to 1. If we sum the intersection (i.e. the dot-product) of e_i with each of the positive examples in the instance, we simply get the total number of ones in the positive examples over these t_i variables. As each variable is set to 1 between $B/2$ and B times, this sum is $B't_i$ for some $B' \in [B/2, B]$. Therefore, the expected intersection of e_i with a random positive example is $\frac{1}{m^+} \cdot t_i B'$. Set $\theta_i = \beta \cdot \frac{t_i B'}{m^+}$, where $\beta > 1$ will be chosen later suitably. Throw out any example which has more than θ_i intersection with e_i . Using Markov’s inequality, we deduce that we retain at least $m^+(1 - \frac{1}{\beta})$ positive examples.

The key point of the above is that focusing on the examples that remain, none of them can contribute more than θ_i hinge-loss (attribute errors), restricting c^{opt} to the t_i variables set to 1 by e_i . On the other hand, it is possible that the number of *actual* errors over positives has increased substantially: perhaps too few of the remaining positive examples share relevant variables with e_i in order for any of the (e_i, θ_i) restricted hypotheses to be a weak learner. We now argue that this cannot happen simultaneously for all e_i .

Specifically, assume for contradiction that none of the (e_i, θ_i) -restricted base hypotheses yields a weak learner. Consider the total number of 1s contributed by the remaining negative examples over the relevant variables of e_i (the relevant variables that are set to 1 by e_i). As each bad negative contributes at most θ_i such ones, the overall contribution on the negative side is $\leq \theta_i \cdot m\text{OPT}(1 + o(1)) = \beta \frac{t_i B'}{m^+} \cdot m\text{OPT}(1 + o(1))$. Since none of relevant variables set to 1 by e_i gives a weak learner, it holds that the number of 1s over the positive side of these relevant variables is no more than $2\beta \frac{m^-}{m^+} \cdot t_i B \cdot \text{OPT}$ (see below, at the specification of the value of γ). So even if each occurrence of 1 comes from a unique positive example, we still have no more

than $2\beta \frac{m}{m^+} \cdot t_i B \cdot \text{OPT}$ positive examples from the (e_i, θ_i) restricted domain intersecting e_i over the relevant variables. Therefore, adding back in the positive examples *not* from the restricted domain, we have no more than $2\beta \frac{m}{m^+} \cdot t_i B \cdot \text{OPT} + m^+/\beta$ positive examples that intersect e_i over the relevant variables.

Consider now a bipartite graph with the \tilde{m} good positive examples on one side and the $m\text{OPT}$ bad negative examples on the other side, with an edge between positive e_j and negative e_i if e_j intersects e_i over the relevant variables. Since each e_i has degree at most $2\beta \frac{m}{m^+} \cdot t_i B \cdot \text{OPT} + m^+/\beta$, the total number of edges is at most $2\beta \frac{m}{m^+} B \text{OPT} \sum_i t_i + m^+ \cdot m\text{OPT}/\beta$, and therefore some good positive examples must have degree at most $\text{OPT} \left[\frac{2\beta B m}{\tilde{m} m^+} \sum_i t_i + \frac{m^+}{\beta} \cdot \frac{m}{\tilde{m}} \right]$. On the other hand, since we are given a (B, α, \tilde{m}) -sparse instance, we know that every good positive example intersects *at least* $\frac{B(1-o(1))}{2}$ negative examples, and moreover that $\sum_i t_i \leq n^{1+\alpha} B \text{OPT}$. Putting this together we have:

$$B/2 \leq (1+o(1))\text{OPT} \left[\frac{2\beta B^2 n^{1+\alpha} \text{OPT} m}{\tilde{m} m^+} + \frac{m^+}{\beta} \cdot \frac{m}{\tilde{m}} \right].$$

Setting $\beta = \sqrt{\frac{(m^+)^2}{2B^2 n^{1+\alpha} \text{OPT}}}$ to equalize the two terms in the sum above, we derive

$$B \leq 4\sqrt{2}(1+o(1))B \cdot \frac{m}{\tilde{m}} \cdot n^{(1+\alpha)/2} \text{OPT}^{3/2}.$$

Thus we have $n^{1+\alpha} \cdot \frac{m^2}{\tilde{m}^2} \cdot \text{OPT}^3 \geq \frac{1+o(1)}{32}$. Recall that $\tilde{m}/m \geq n^{-o(\alpha)}$, so we derive a contradiction, as for sufficiently large n it must hold that

$$\text{OPT} \geq \left(\frac{1+o(1)}{32} \right)^{1/3} n^{-\frac{1+\alpha}{3}-o(\alpha)} > n^{-1/3-\alpha}.$$

In order to complete the proof, we need to verify that indeed $\beta > 1$. Recall $B = O(m\text{OPT})$ and $m^+ \geq \tilde{m}$, so $m^+/m \geq n^{-o(\alpha)}$. Thus $\beta^2 = \Omega\left(\frac{1}{n^{1+\alpha+o(\alpha)} \text{OPT}^3}\right) = \Omega(n^{2\alpha-o(\alpha)})$ by our assumption on OPT .

The last detail is to check what advantage do we get over a random guess. Our analysis shows that for some bad negative example e_i , the number of ones over the relevant variables on the positive side is at least $2\beta \frac{m}{m^+} \cdot t_i B \cdot \text{OPT}$, whereas on the negative side, there can be at most $\beta \frac{m}{m^+} \cdot t_i B \cdot \text{OPT}(1+o(1))$ ones. We deduce that at least one of the at most $\min(r, t_i)$ relevant variables set to 1 by e_i must give a gap of at least $\frac{\beta \cdot t_i B m \cdot \text{OPT}(1-o(1))}{m^+ \min(r, t_i)} > B \cdot \text{OPT}(1-o(1))$ since $\beta > 1$. Finally, using the fact that $B = \Omega(\tilde{m}/r)$ we get a gap of $\Omega\left(\frac{\tilde{m}\text{OPT}}{r}\right)$ or equivalently an advantage of $\gamma = \Omega\left(\frac{\text{OPT}}{r} \cdot \frac{\tilde{m}}{m}\right)$. This advantage is trivially $\Omega(n^{-2(1+o(\alpha))})$, or, using the assumption $\text{OPT} > 1/\sqrt{n}$ (for otherwise, we can apply Peleg's algorithm [14]), we get $\gamma = \Omega(n^{-\frac{3}{2}(1+o(\alpha))})$. \blacksquare

3.2 General Instances

Section 3.1 dealt with nicely behaved (homogeneous) instances. In order to complete the proof of Theorem 2, we need to show how to reduce a general instance to such a (B, α, \tilde{m}) -sparse instance. What we show is a (simple) algorithm that partitions a given instance into sub-instances, based on the number of 1s of each example over certain variables (but without looking at the labels of the examples). It outputs a $\text{polylog}(n)$ -long list of sub-instances, each containing a noticeable fraction of the domain, and has the following guarantee: either some sub-instance has a trivial weak-learner (has a noticeably different number of positive versus negative examples or there is a variable with noticeable correlation), or some sub-instance is (B, α, \tilde{m}) -sparse. Formally, we prove this next lemma.

Lemma 5 *There exists a $\text{poly}((\log n)^{O(1/\alpha)}, n, m)$ -time algorithm, that gets as an input $2m$ labeled examples in $\{0, 1\}^n$, and output a list of subsets, each containing $m/\text{polylog}(n)$ examples, s.t. either some subset has a trivial weak-learner, or some subset is $(B, \alpha, m/\text{polylog}(n))$ -sparse.*

Combining the algorithm from Lemma 5 with the algorithm presented in Section 3.1, we get our weak-learning algorithm (see Algorithm 2). We first run the algorithm of Lemma 5, traverse all sub-instances, and check whether any has a trivial weak-learner. If not, we run the algorithm for (B, α, \tilde{m}) -sparse instances over each sub-instance. Obviously, given the one sub-instance which is sparse, we find a restricted hypothesis with $\tilde{\Omega}(n^{-2})$ advantage over a random guess.

Proof: We start by repeating the argument presented in the introduction (Section 1.1). For any relevant variable, no more than $m_{\text{bad}} \leq m \cdot \text{OPT}(1+o(1))$ bad examples set it to 1. Therefore, as an initial step, we throw out any variable with more than this many occurrences over the negative examples, as it cannot

possibly be a relevant variable. For convenience, redefine n to be the number of variables that remain. Next, we check each individual variable to determine if it itself is a weak predictor. If not, then this means each variable is set to 1 on approximately the same number of positive and negative examples.

Bucket all the variables according the number of times they are set to 1, where the j -bucket contains all the variables that are set to 1 any number of times in the range $[2^j, 2^{j+1})$. Since there are at most $\log n$ buckets, some bucket j must cover at least $\frac{m^+}{\log n}$ positive examples, in the sense that the disjunction over the *relevant* variables in this bucket agrees with at least this many good positives. So now, let $B' = 2^{j+1}$, let n' and r' be the total number of variables and the number of relevant variables in this bucket respectively. As we can ignore all examples that are identically 0 over the n' variables in this bucket, let m'^+ (resp. m'^-) be the number of positive (resp. negative) examples covered by the variables in this bucket. Our algorithm adds the remaining examples (over these n' variables) as one sub-instance to its list. Let the number of these examples be $2m'$. As before, if the number of positive examples and negative examples covered by these n' variables differ significantly, or if some variable is a weak learner (with respect to the set of examples left), then the algorithm halts. Observe that if this sub-instance is $(B', \alpha, m/\log(n))$ -sparse, then we are done, no matter what other sub-instances the algorithm will add to its list.

Focusing on the remaining examples, every variable is set to 1 at most B' many times over the positive examples, so the total number of 1s, over the positive examples is $\leq n'B'$. If indeed the resulting instance is not $(B', \alpha, m/\log(n))$ -sparse, then the total number of 1s over the bad negative examples is $\geq (n')^{1+\alpha}(B')\text{OPT}$. So now, our algorithm throws out any example with more than $2n'B'/m'$ variables set to 1, and adds the remaining examples to the list of sub-instances. By Markov's inequality, we are guaranteed not to remove more than $1/2$ of the positive examples, so the sub-instance remaining is sufficiently large. As before, if the remaining subset of examples (over these n' variables) has a trivial weak-learner, we are done. Otherwise, the algorithm continues recursively over this sub-instance – re-buckets and then removes all examples with too many variables set to 1. Note, each time the algorithm buckets the variables, it needs to recurse over each bucket that covers at least a $1/\log(n)$ fraction of the positive examples. In the worst-case, all of the $\log(n)$ buckets cover these many positive examples, and therefore, the branching factor in each bucketing step is $\log(n)$.

We now show that the depth of the bucket-and-remove recurrence is no more than $O(1/\alpha)$. It is easy to see inductively that at the i -th step of the recursion, we retain a fraction of $m/(\log n)^i$ positive examples. Suppose that by the first i steps, no sub-instance is sparse and no weak-learner is found. Recall, if $r\text{OPT} \ll 1$, we have an immediate weak-learner, so it must hold that in the i -th step, we still retain at least $n_i = 1/\text{OPT}$ variables. Furthermore, as in the i -th step we did not have a sparse instance, it follows that the bad negative examples had more than $(n_i)^{1+\alpha}(B')\text{OPT}$ ones before we threw out examples. Once we remove dense examples, they contain no more than $\frac{2(n_i)(B')}{m_i} \cdot m\text{OPT}$ many ones. Thus, the fraction of ones over the bad negatives that survive each removal step is no more than $n_i^{-\alpha} \cdot \frac{m}{m_i}$. As $1/\text{OPT} > n^{1/3}$, this fraction is at most $n^{-\alpha/3}(\log n)^i < n^{-\alpha/6}$ (for the first $O(1/\alpha)$ iterations). Hence, after $6/\alpha$ iterations, some relevant variable must be a weak-learner.

To complete the proof, note that we take no more than $(\log n)^{6/\alpha}$ bucket-and-remove steps. Each such step requires $\text{poly}(n, m)$ time for the bucketing, removal and checking for weak-learner. We conclude that the run-time of this algorithm is $\text{poly}((\log n)^{1/\alpha}, n, m)$. ■

Algorithm 2 A weak learner for general instances.

Input: A set of $2m$ training examples.

Step 1: If any individual variable or the constant hypotheses is a weak learner, output it and halt.

Step 2: Remove any variable which has more than $2m\text{OPT}$ 1's over the negative examples.

Step 3: Bucket the remaining variables such that bucket j contains variables with density in $[2^j, 2^{j+1})$.

Step 4: For every bucket which covers at least a $\log n$ fraction of the positive examples

Step 4a: Run the algorithm for sparse instances on this bucket. If a weak learner is obtained, output it and halt.

Step 4b: Let B' be the density (2^{j+1}) in this bucket, n' be the number of variables in the bucket and $2m'$ be the total number of examples with respect to this bucket (ignoring the ones which are identically zero over the n' variables). Remove all the examples which have more than $2n'B'/m'$ 1's over this bucket. Repeat steps 1-4 on this new instance.

3.3 Strong Learning

Given Theorem 2, we now prove the main theorem (Theorem 1) by plugging the weak-learner into an off-the-shelf boosting algorithm for the agnostic case. We use the recent `ABOOSTDI` booster of [4], which converts any algorithm satisfying Theorem 2 into one satisfying Theorem 1. The result in [4] gives a boosting technique for (η, γ) -weak learners. In our context an (η, γ) -weak learner is an algorithm which with respect to any distribution \mathcal{D} , with high probability, produces a hypothesis of error $\leq \frac{1}{2} - \gamma$, whenever $\text{OPT}_{\text{disj}} \leq \frac{1}{2} - \eta$.

Theorem 6 (Feldman [4], Theorem 3.5) *There exists an algorithm `ABOOSTDI` that, given a (η, γ) -weak learner, for every distribution \mathcal{D} and $\epsilon > 0$, produces, with high probability, a hypothesis h such that $\text{err}_{\mathcal{D}}(h) \leq \frac{\text{OPT}_{\text{disj}}}{1-2\eta} + \epsilon$. Furthermore, the running time of the algorithm is $T \cdot \text{poly}(\frac{1}{\gamma}, \frac{1}{\epsilon})$, where T is the running time of the weak learner.*

As an immediate corollary, we set $\eta = \frac{1}{2} - \frac{1}{2} \cdot n^{-1/3-\alpha}$ and obtain an hypothesis h such that $\text{err}_{\mathcal{D}}(h) \leq 2n^{1/3+\alpha}\text{OPT} + \epsilon$. This concludes the proof of Theorem 1. We note that as an alternative to `ABOOSTDI`, one can also use the boosting algorithm of Kalai et. al [9], followed by another boosting algorithm of Gavinsky [7], to get the result in Theorem 1.

4 Future Directions

In this paper we have presented an algorithm for learning the class of disjunctions in the case that $\text{OPT} < n^{-(1/3+\alpha)}$, achieving an error rate of $O(n^{1/3+\alpha} \cdot \text{OPT}) + \epsilon$. The natural open question is whether one can improve this bound. For example, can one achieve weak agnostic learning for $\text{OPT} = n^{-1/4}$? Or, can one improve the bounds as a function of the number of relevant variables, e.g., making only a factor $O(r^{0.9})$ times more mistakes than the best disjunction?

An intriguing open question is whether one can extend this technique for other concept classes. For example, consider the class of linear separators over $\{0, 1\}^n$ with weights in $\{0, 1\}$ (i.e., majority vote or “ k of r ” functions). Here we do not know even how to achieve weak learning for $\text{OPT} = n^{-0.99}$. The algorithm presented in this paper for disjunctions uses the fact that in order for individual variables not to be weak hypotheses themselves, the bad negative examples must in some sense “point” in the direction of the target vector (they must have a high dot-product with the target function vector if we view the disjunction as a linear threshold function) to a substantially greater extent than the positive examples do. E.g., if a typical positive example has t relevant variables set to 1, then the typical bad negative example must have t/OPT relevant variables set to 1. For the case of majority-vote functions, the difficulty with this approach is that instead all we can say is that if the positive examples have $r/2 + t$ relevant variables set to 1, then the typical bad negative examples should have at least $r/2 + t/\text{OPT}$ relevant variables set to 1, which might not be such a distinction in a multiplicative sense.

On a more general note, our work here uses somewhat non-traditional hypotheses, by using the examples themselves to define “slices” of the data (focusing on those examples with no more than a certain θ dot-product with some given negative example). Perhaps this might be useful for other learning problems.

Acknowledgments

We would like to thank Aaron Roth and Maria-Florina Balcan for a number of helpful discussions.

References

- [1] Peter Auer and Manfred K. Warmuth. Tracking the best disjunction. In *Proc. 36th Symposium on Foundations of Computer Science*, pages 312–321, 1995.
- [2] R.D. Carr, S. Doddi, G. Konjevod, and M. Marathe. On the red-blue set cover problem. In *Proceedings of the Eleventh Annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 345–353, 2000.
- [3] Vitaly Feldman. Optimal hardness results for maximizing agreements with monomials. *SICOMP*, 39(2), 2009. Extended abstract in CCC 2006.
- [4] Vitaly Feldman. Distribution-specific agnostic boosting. In *1st Symposium on Innovations in Computer Science (ICS)*, pages 241–250, 2010.
- [5] Vitaly Feldman, Venkatesan Guruswami, Prasad Raghavendra, and Yi Wu. Agnostic learning of monomials by halfspaces is hard. In *FOCS*, 2009.
- [6] Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Mach. Learn.*, 21(3):269–304, 1995.

- [7] Dmitry Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *J. Mach. Learn. Res.*, 4:101–117, 2003.
- [8] A. Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008.
- [9] Adam Tauman Kalai, Yishay Mansour, and Elad Verbin. On agnostic boosting and parity learning. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 629–638, New York, NY, USA, 2008. ACM.
- [10] A. Klivans, R. O'Donnell, and R. Servedio. Learning geometric concepts via gaussian surface area. In *FOCS*, 2008.
- [11] Adam R. Klivans, Philip M. Long, and Rocco A. Servedio. Learning halfspaces with malicious noise. In *ICALP*, 2009.
- [12] N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. In *Proc. 4th Conference on Computational Learning Theory*, pages 147–156, Santa Cruz, California, 1991. Morgan Kaufmann.
- [13] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4), 1987.
- [14] David Peleg. Approximation algorithms for the label-covermax and red-blue set cover problems. *J. of Discrete Algorithms*, 5(1):55–64, 2007.

Mansour's Conjecture is True for Random DNF Formulas

Adam Klivans

University of Texas at Austin
klivans@cs.utexas.edu

Homin K. Lee

University of Texas at Austin
homin@cs.utexas.edu

Andrew Wan

Columbia University
atw12@cs.columbia.edu

Abstract

In 1994, Y. Mansour conjectured that for every DNF formula on n variables with t terms there exists a polynomial p with $t^{O(\log(1/\epsilon))}$ non-zero coefficients such that $\mathbf{E}_{x \in \{0,1\}^n} [(p(x) - f(x))^2] \leq \epsilon$. We make the first progress on this conjecture and show that it is true for several natural subclasses of DNF formulas including randomly chosen DNF formulas and read- k DNF formulas for constant k .

Our result yields the first polynomial-time query algorithm for agnostically learning these subclasses of DNF formulas with respect to the uniform distribution on $\{0,1\}^n$ (for any constant error parameter).

Applying recent work on sandwiching polynomials, our results imply that a $t^{-O(\log 1/\epsilon)}$ -biased distribution fools the above subclasses of DNF formulas. This gives pseudorandom generators for these subclasses with shorter seed length than all previous work.

1 Introduction

Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a DNF formula, *i.e.*, a function of the form $T_1 \vee \dots \vee T_t$ where each T_i is a conjunction of at most n literals. In this paper we are concerned with the following question: How well can a real-valued polynomial p approximate the Boolean function f ? This is an important problem in computational learning theory, as real-valued polynomials play a critical role in developing learning algorithms for DNF formulas.

Over the last twenty years, considerable work has gone into finding polynomials p with certain properties (*e.g.*, low-degree, sparse) such that

$$\mathbf{E}_{x \in \{0,1\}^n} [(p(x) - f(x))^2] \leq \epsilon.$$

In 1989, Linial et al. (1993) were the first to prove that for any t -term DNF formula f , there exists a polynomial $p : \{0,1\}^n \rightarrow \mathbb{R}$ of degree $O(\log(t/\epsilon)^2)$ such that $\mathbf{E}_{x \in \{0,1\}^n} [(p(x) - f(x))^2] \leq \epsilon$. They showed that this type of approximation implies a quasipolynomial-time algorithm for PAC learning DNF formulas with respect to the uniform distribution. Kalai et al. (2008) observed that this fact actually implies something stronger, namely a quasipolynomial-time agnostic learning algorithm for learning DNF formulas (with respect to the uniform distribution). Additionally, the above approximation was used in recent work due to Bazzi (2007) and Razborov (2008) to show that bounded independence fools DNF formulas.

Three years later, building on the work of Linial et al. (1993) Mansour (1995) proved that for any t -term DNF formula, there exists a polynomial p defined over $\{0,1\}^n$ with *sparsity* $t^{O(\log \log t \log(1/\epsilon))}$ such that $\mathbf{E}_{x \in \{0,1\}^n} [(p(x) - f(x))^2] \leq \epsilon$ (for $1/\epsilon = \text{poly}(n)$). By sparsity we mean the number of non-zero Fourier coefficients of p . This result implied a nearly polynomial-time *query* algorithm for PAC learning DNF formulas with respect to the uniform distribution.

Mansour conjectured (Mansour, 1994) that the above bound could be improved to $t^{O(\log 1/\epsilon)}$. Such an improvement would imply a polynomial-time query algorithm for learning DNF formulas with respect to the uniform distribution (to within any constant accuracy), and learning DNF formulas in this model was a major open problem at that time.

In a celebrated work from 1994, Jeff Jackson proved that DNF formulas were learnable in polynomial time (with queries, with respect to the uniform distribution) *without* proving the Mansour conjecture. His

“Harmonic Sieve” algorithm (Jackson, 1997) used boosting in combination with some weak approximation properties of polynomials. As such, for several years, Mansour’s conjecture remained open and attracted considerable interest, but its resolution did not imply any new results in learning theory.

In 2008, Gopalan et al. (2008b) proved that a positive resolution to the Mansour conjecture also implies an efficient query algorithm for *agnostically* learning DNF formulas (to within any constant error parameter). The agnostic model of learning is a challenging learning scenario that requires the learner to succeed in the presence of adversarial noise. Roughly, Gopalan et al. (2008b) showed that if a class of Boolean functions \mathcal{C} can be ϵ -approximated by polynomials of sparsity s , then there is a query algorithm for agnostically learning \mathcal{C} in time $\text{poly}(s, 1/\epsilon)$ (since decision trees are approximated by sparse polynomials, they obtained the first query algorithm for agnostically learning decision trees with respect to the uniform distribution on $\{0, 1\}^n$). Whether DNF formulas can be agnostically learned (with queries, with respect to the uniform distribution) still remains a difficult open problem (Gopalan et al., 2008a).

1.1 Our Results

We prove that the Mansour conjecture is true for several well-studied subclasses of DNF formulas. As far as we know, prior to this work, the Mansour conjecture was not known to be true for any interesting class of DNF formulas.

Our first result shows that the Mansour conjecture is true for the class of randomly chosen DNF formulas:

Theorem 1 *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a DNF formula with $t = n^{O(1)}$ terms where each term is chosen independently from the set of all terms of length $\lfloor \log t \rfloor$. Then with probability $1 - n^{-\Omega(1)}$ (over the choice of the DNF formula), there exists a polynomial p with sparsity $t^{O(\log 1/\epsilon)}$ such that $\mathbf{E}[(p(x) - f(x))^2] \leq \epsilon$.*

For $t = n^{\Theta(1)}$, the conclusion of Theorem 1 holds with probability at least $1 - n^{-\Omega(\log t)}$. Our second result is that the Mansour conjecture is true for the class of read- k DNF formulas:

Theorem 2 *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a DNF formula with t terms where each literal appears at most k times. Then there exists a polynomial p with sparsity $t^{O(16^k \log 1/\epsilon)}$ such that $\mathbf{E}[(p(x) - f(x))^2] \leq \epsilon$.*

Even for the case $k = 1$, Mansour’s conjecture was not known to be true. Mansour (1995) proves that any polynomial that approximates read-once DNF formulas to ϵ accuracy must have *degree* at least $d = \Omega(\log t \log(1/\epsilon) / \log \log(1/\epsilon))$. He further shows that a “low-degree” strategy of selecting all of a DNF formula’s Fourier coefficients of monomials up to degree d results in a polynomial p with sparsity $t^{O(\log \log t \log 1/\epsilon)}$ for $1/\epsilon = \text{poly } n$. It is not clear, however, how to improve this to the desired $t^{O(\log 1/\epsilon)}$ bound.

As mentioned earlier, by applying the result of Gopalan et al. (2008b), we obtain the first polynomial-time query algorithms for agnostically learning the above classes of DNF formulas to within any constant accuracy parameter. We consider this an important step towards agnostically learning all DNF formulas.

Corollary 3 *Let \mathcal{C} be the class of DNF formulas with $t = n^{O(1)}$ terms where each term is randomly chosen from the set of all terms of length $\lfloor \log t \rfloor$. Then there is a query-algorithm for agnostically learning \mathcal{C} with respect to the uniform distribution on $\{0, 1\}^n$ to accuracy ϵ in time $\text{poly}(n) \cdot t^{O(\log 1/\epsilon)}$ with probability $1 - n^{-\Omega(1)}$ (over the choice of the DNF formula).*

We define the notion of agnostic learning with respect to randomly chosen concept classes in Section 2. For $t = n^{\Theta(1)}$, Corollary 3 holds for a $1 - n^{-\Omega(\log t)}$ fraction of randomly chosen DNF formulas. We also obtain a corresponding agnostic learning algorithm for read- k DNF formulas:

Corollary 4 *Let \mathcal{C} be the class of read- k DNF formulas with t terms. Then there is a query-algorithm for agnostically learning \mathcal{C} with respect to the uniform distribution on $\{0, 1\}^n$ to accuracy ϵ in time $\text{poly}(n) \cdot t^{O(16^k \log 1/\epsilon)}$.*

Our sparse polynomial approximators can also be used in conjunction with recent work due to De et al. (2009) to show that for randomly chosen polynomial-size DNF formulas or read- k DNF formulas f , a $t^{-O(\log 1/\epsilon)}$ -biased distribution fools f (for $k = O(1)$):

Theorem 5 *Let f be a randomly chosen polynomial-size DNF formula or a read- k DNF formula. Then (with probability $1 - n^{-\Omega(1)}$ for random DNF formulas) there exists a pseudorandom generator G such that*

$$\left| \Pr_{x \in \{0, 1\}^s} [f(G(x)) = 1] - \Pr_{z \in \{0, 1\}^n} [f(z) = 1] \right| \leq \epsilon$$

with $s = O(\log n + \log t \cdot \log(1/\epsilon))$.

Previously it was only known that these types of biased distributions fool read-once DNF formulas (De et al., 2009).

1.2 Related Work

As mentioned earlier, Mansour, using the random restriction machinery of Håstad (1986) and Linial et al. (1993) had shown that for any DNF formula f , there exists a polynomial of sparsity $t^{O(\log \log t \log 1/\epsilon)}$ that approximates f .

The subclasses of DNF formulas that we show are agnostically learnable have been well-studied in the PAC model of learning. Monotone read- k DNF formulas were shown to be PAC-learnable with respect to the uniform distribution by Hancock and Mansour (1991), and random DNF formulas were recently shown to be learnable on average with respect to the uniform distribution in the following sequence of work (Jackson & Servedio, 2005; Jackson et al., 2008; Sellie, 2008; Sellie, 2009).

Recently (and independently) De et al. (2009) proved that for any read-once DNF formula f , there exists an approximating polynomial p of sparsity $t^{O(\log 1/\epsilon)}$. More specifically, De et al. (2009) showed that for any class of functions \mathcal{C} fooled by δ -biased sets, there exist sparse, sandwiching polynomials for \mathcal{C} where the sparsity depends on δ . Since they show that $t^{-O(\log 1/\epsilon)}$ -biased sets fool read-once DNF formulas, the existence of a sparse approximator for the read-once case is implicit in their work.

1.3 Our Approach

As stated above, our proof does not analyze the Fourier coefficients of DNF formulas, and our approach is considerably simpler than the random-restriction method taken by Mansour (we consider the lack of Fourier analysis a feature of the proof, given that all previous work on this problem has been Fourier-based). Instead, we use polynomial interpolation.

A Basic Example. Consider a DNF formula $f = T_1 \vee \dots \vee T_t$ where each T_i is on a disjoint set of exactly $\log t$ variables (assume t is a power of 2). The probability that each term is satisfied is $1/t$, and the expected number of satisfied terms is one. Further, since the terms are disjoint, with high probability over the choice of the random input, only a few—say d —terms will be satisfied. As such, we construct a univariate polynomial p with $p(0) = 0$ and $p(i) = 1$ for $1 \leq i \leq d$. Then $p(T_1 + \dots + T_t)$ will be exactly equal to f as long as at most d terms are satisfied. A careful calculation shows that the inputs where p is incorrect will not contribute too much to $\mathbf{E}[(f - p)^2]$, as there are few of them. Setting parameters appropriately yields a polynomial p that is both sparse and an ϵ -approximator of f .

Random and read-once DNF formulas. More generally, we adopt the following strategy: given a DNF formula f (randomly chosen or read-once) either (1) with sufficiently high probability a random input does not satisfy too many terms of f or (2) f is highly biased. In the former case we can use polynomial interpolation to construct a sparse approximator and in the latter case we can simply use the constant 0 or 1 function.

The probability calculations are a bit delicate, as we must ensure that the probability of many terms being satisfied decays faster than the growth rate of our polynomial approximators. For the case of random DNF formulas, we make use of some recent work due to Jackson et al. (2008) on learning random monotone DNF formulas.

Read- k DNF formulas. Read- k DNF formulas do not fit into the above dichotomy so we do not use the sum $T_1 + \dots + T_t$ inside the univariate polynomial. Instead, we use a sum of *formulas* (rather than terms) based on a construction from (Razborov, 2008). We modify Razborov's construction to exploit the fact that terms in a read- k DNF formula do not share variables with many other terms. Our analysis shows that we can then employ the previous strategy: either (1) with sufficiently high probability a random input does not satisfy too many formulas in the sum or (2) f is highly biased.

2 Preliminaries

In this paper, we will primarily be concerned with Boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$. Let x_1, \dots, x_n be Boolean variables. A *literal* is either a variable x_i or its negation \bar{x}_i , and a *term* is a conjunction of literals. Any Boolean function can be expressed as a disjunction of terms, and such a formula is said to be a *disjunctive normal form* (or DNF) formula. A read- k DNF formula is a DNF formula in which the maximum number of occurrences of each variable is bounded by k . A Boolean function is monotone if changing the value of an input bit from 0 to 1 never causes the value of f to change from 1 to 0. The following consequence (Kleitman, 1966; Alon & Spencer, 2000) of the Four Functions Theorem will be useful in our study of monotone functions.

Theorem 6 Let $e, f, \neg g$, and $\neg h$ be monotone Boolean functions over $\{0, 1\}^n$. Then for any product distribution \mathcal{D} over $\{0, 1\}^n$, $\Pr_{\mathcal{D}}[e \wedge f] \geq \Pr_{\mathcal{D}}[e] \Pr_{\mathcal{D}}[f]$, $\Pr_{\mathcal{D}}[g \wedge h] \geq \Pr_{\mathcal{D}}[g] \Pr_{\mathcal{D}}[h]$, and $\Pr_{\mathcal{D}}[f \wedge g] \leq \Pr_{\mathcal{D}}[f] \Pr_{\mathcal{D}}[g]$.

2.1 Sparse Polynomials

Every function $f : \{0, 1\}^n \rightarrow \mathbb{R}$ can be expressed by its Fourier expansion: $f(x) = \sum_S \hat{f}(S) \chi_S(x)$ where $\chi_S(x) = \prod_{i \in S} (-1)^{x_i}$ for $S \subseteq [n]$, and $\hat{f}(S) = \mathbf{E}[f \cdot \chi_S]$. The Fourier expansion of f can be thought of as the unique polynomial representation of f over $\{+1, -1\}^n$ under the map $x_i \mapsto 1 - 2x_i$.

Mansour conjectured that polynomial-size DNF formulas could be approximated by *sparse* polynomials over $\{+1, -1\}^n$. We say a polynomial $p : \{+1, -1\}^n \rightarrow \mathbb{R}$ has sparsity s if it has at most s non-zero coefficients. We state Mansour's conjecture as originally posed in (Mansour, 1994), which uses the convention of representing FALSE by $+1$ and TRUE by -1 .

Conjecture 7 (Mansour, 1994) Let $f : \{+1, -1\}^n \rightarrow \{+1, -1\}$ be any function computable by a t -term DNF formula. Then there exists a polynomial $p : \{+1, -1\}^n \rightarrow \mathbb{R}$ with $t^{O(\log 1/\epsilon)}$ terms such that $\mathbf{E}[(f - p)^2] \leq \epsilon$.

We will prove the conjecture to be true for various subclasses of polynomial-size DNF formulas. In our setting, Boolean functions will output 0 for FALSE and 1 for TRUE. However, we can easily change the range by setting $f^\pm := 1 - 2 \cdot f$. Changing the range to $\{+1, -1\}$ changes the accuracy of the approximation by at most a factor of 4: $\mathbf{E}[(1 - 2f) - (1 - 2p)]^2 = 4 \mathbf{E}[(f - p)^2]$, and it increases the sparsity by at most 1.

Given a Boolean function f , we construct a sparse approximating polynomial over $\{+1, -1\}^n$ by giving an approximating polynomial $p : \{0, 1\}^n \rightarrow \mathbb{R}$ with real coefficients that has small spectral norm. The rest of the section gives us some tools to construct such polynomials and explains why doing so yields sparse approximators.

Definition 8 The Fourier ℓ_1 -norm (also called the spectral norm) of a function $p : \{0, 1\}^n \rightarrow \mathbb{R}$ is defined to be $\|p\|_1 := \sum_S |\hat{p}(S)|$. We will also use the following minor variant, $\|p\|_1^{\neq 0} := \sum_{S \neq \emptyset} |\hat{p}(S)|$.

The following two facts about the spectral norm of functions will allow us to construct polynomials over $\{0, 1\}^n$ naturally from DNF formulas.

Fact 9 Let $p : \{0, 1\}^m \rightarrow \mathbb{R}$ be a polynomial with coefficients $p_S \in \mathbb{R}$ for $S \subseteq [m]$, and $q_1, \dots, q_m : \{0, 1\}^n \rightarrow \{0, 1\}$ be arbitrary Boolean functions. Then $p(q_1, \dots, q_m) = \sum_S p_S \prod_{i \in S} q_i$ is a polynomial over $\{0, 1\}^n$ with spectral norm at most

$$\sum_{S \subseteq [m]} |p_S| \prod_{i \in S} \|q_i\|_1.$$

Proof: The fact follows by observing that for any $p, q : \{0, 1\}^n \rightarrow \mathbb{R}$, we have $\|p + q\|_1 \leq \|p\|_1 + \|q\|_1$ and $\|pq\|_1 \leq \|p\|_1 \|q\|_1$. ■

Fact 10 Let $T : \{0, 1\}^n \rightarrow \{0, 1\}$ be an AND of a subset of its literals. Then $\|T\|_1 = 1$.

Finally, using the fact below, we show why approximating polynomials with small spectral norm give sparse approximating polynomials.

Fact 11 (Kushilevitz & Mansour, 1993) Given any function $f : \{0, 1\}^n \rightarrow \mathbb{R}$ and $\epsilon > 0$, let $\mathcal{S} = \{S \subseteq [n] : |\hat{f}(S)| \geq \epsilon / \|f\|_1\}$, and $g(x) = \sum_{S \in \mathcal{S}} \hat{f}(S) \chi_S(x)$. Then $\mathbf{E}[(f - g)^2] \leq \epsilon$, and $|\mathcal{S}| \leq \|f\|_1^2 / \epsilon$.

Now, given functions $f, p : \{0, 1\}^n \rightarrow \mathbb{R}$ such that $\mathbf{E}[(f - p)^2] \leq \epsilon$, we can construct a 4ϵ -approximator for f with sparsity $\|p\|_1^2 / \epsilon$ by defining $p'(x) = \sum_{S \in \mathcal{S}} \hat{p}(S) \chi_S(x)$ as in Fact 11. Clearly p' has sparsity $\|p\|_1^2 / \epsilon$, and

$$\mathbf{E}[(f - p')^2] = \mathbf{E}[(f - p + p - p')^2] \leq \mathbf{E}[2((f - p)^2 + (p - p')^2)] \leq 4\epsilon,$$

where the first inequality follows from the inequality $(a + b)^2 \leq 2(a^2 + b^2)$ for any reals a and b .

2.2 Agnostic learning

We first describe the traditional framework for agnostically learning concept classes with respect to the uniform distribution and then give a slightly modified definition for an “average-case” version of agnostic learning where the unknown concept (in this case a DNF formula) is randomly chosen.

Definition 12 (Standard agnostic model) Let $f : \{+1, -1\}^n \rightarrow \{+1, -1\}$ be an arbitrary function, and let \mathcal{D} be the uniform distribution on $\{+1, -1\}^n$. Define

$$\text{opt} = \min_{c \in \mathcal{C}} \Pr_{x \sim \mathcal{D}} [c(x) \neq f(x)].$$

That is, opt is the error of the best fitting concept in \mathcal{C} with respect to \mathcal{D} . We say that an algorithm A agnostically learns \mathcal{C} with respect to \mathcal{D} if the following holds for any f : if A is given black-box access to f then with high probability A outputs a hypothesis h such that $\Pr_{x \sim \mathcal{D}} [h(x) \neq f(x)] \leq \text{opt} + \epsilon$.

The intuition behind the above definition is that a learner—given access to a concept $c \in \mathcal{C}$ where an η fraction of c 's inputs have been adversarially corrupted—should still be able to output a hypothesis with accuracy $\eta + \epsilon$ (achieving error better than η may not be possible, as the adversary could embed a completely random function on an η fraction of c 's inputs). Here η plays the role of opt .

This motivates the following definition for agnostically learning a randomly chosen concept from some class \mathcal{C} :

Definition 13 (Agnostically learning random concepts) Let \mathcal{C} be a concept class and choose c randomly from \mathcal{C} (the distribution over \mathcal{C} will be clear from the context). We say that an algorithm A agnostically learns random concepts from \mathcal{C} if with probability at least $1 - \delta$ over the choice of c the following holds: if the learner is given black-box access to some fixed function c' and $\Pr_{x \in \{+1, -1\}^n} [c(x) \neq c'(x)] \leq \eta$, then A outputs a hypothesis h such that $\Pr_{x \in \{+1, -1\}^n} [h(x) \neq c'(x)] \leq \eta + \epsilon$.

We are unaware of any prior work defining an agnostic framework for learning randomly chosen concepts.

The main result we use to connect the approximation of DNF formulas by sparse polynomials with agnostic learning is due to Gopalan et al. (2008b):

Theorem 14 (Gopalan et al., 2008b) Let \mathcal{C} be a concept class such that for every $c \in \mathcal{C}$ there exists a polynomial p such that $\|p\|_1 \leq s$ and $\mathbf{E}_{x \in \{+1, -1\}^n} [|p(x) - c(x)|^2] \leq \epsilon^2/2$. Then there exists an algorithm B such that the following holds: given black-box access to any Boolean function $f : \{+1, -1\}^n \rightarrow \{+1, -1\}$, B runs in time $\text{poly}(n, s, 1/\epsilon)$ and outputs a hypothesis $h : \{+1, -1\}^n \rightarrow \{+1, -1\}$ with

$$\Pr_{x \in \{+1, -1\}^n} [h(x) \neq f(x)] \leq \text{opt} + \epsilon.$$

3 Approximating DNFs using univariate polynomial interpolation

Let $f = T_1 \vee T_2 \vee \dots \vee T_t$ be any DNF formula. We say $T_i(x) = 1$ if x satisfies the term T_i , and 0 otherwise. Let $y_f : \{0, 1\}^n \rightarrow \{0, \dots, t\}$ be the function that outputs the number of terms of f satisfied by x , i.e., $y_f(x) = T_1(x) + T_2(x) + \dots + T_t(x)$.

Our constructions will use the following univariate polynomial P_d to interpolate the values of f on inputs $\{x : y_f(x) \leq d\}$.

Fact 15 Let

$$P_d(y) := (-1)^{d+1} \frac{(y-1)(y-2)\dots(y-d)}{d!} + 1. \quad (1)$$

Then, (1) the polynomial P_d is a degree- d polynomial in y ; (2) $P_d(0) = 0$, $P_d(y) = 1$ for $y \in [d]$, and for $y \in [t] \setminus [d]$, $P_d(y) = -\binom{y-1}{d} + 1 \leq 0$ if d is even and $P_d(y) = \binom{y-1}{d} + 1 > 1$ if d is odd; and (3) the sum of the magnitudes of P_d 's coefficients is d .

Proof: Properties (1) and (2) can be easily verified by inspection. Expanding the falling factorial, we get that $(y-1)(y-2)\dots(y-d) = \sum_{j=0}^d (-1)^{d-j} \binom{d+1}{j+1} y^j$, where $\binom{a}{b}$ denotes a Stirling number of the first kind. The Stirling numbers of the first kind count the number of permutations of a elements with b disjoint cycles. Therefore, $\sum_{j=0}^d \binom{d+1}{j+1} = (d+1)!$ (Graham et al., 1994). The constant coefficient of P_d is 0 by Property (2), thus the sum of the absolute values of the other coefficients is $((d+1)! - d!)/d! = d$. ■

For any t -term DNF formula f , we can construct a polynomial $p_{f,d} : \{0, 1\}^n \rightarrow \mathbb{R}$ defined as $p_{f,d} := P_d \circ y_f$. A simple calculation, given below, shows that the ℓ_1 -norm of $p_{f,d}$ is polynomial in t and exponential in d .

Lemma 16 Let f be a t -term DNF formula, then $\|p_{f,d}\|_1 \leq t^{O(d)}$.

Proof: By Fact 15, P_d is a degree- d univariate polynomial with d non-zero coefficients of magnitude at most d . We can view the polynomial $p_{f,d}$ as the polynomial $P'_d(T_1, \dots, T_t) := P_d(T_1 + \dots + T_t)$ over variables $T_i \in \{0, 1\}$. Expanding out P'_d gives us at most dt^d monomials with coefficients of magnitude at most d . Now each monomial of P'_d is a product of T_i 's, so applying Facts 10 and 9 we have that $\|p_{f,d}\|_1 \leq t^{O(d)}$. ■

The next two sections will show that the polynomial $p_{f,d}$ (for $d = \Theta(\log 1/\epsilon)$) is in fact a good approximation for random DNF formulas and (with a slight modification) for read- k DNF formulas. As a warm-up, we will show the simple case of read-once DNF formulas.

3.1 A Simple Case: Read-Once DNF Formulas

For read-once DNF formulas, the probability that a term is satisfied is independent of whether or not any of the other terms are satisfied, and thus f is unlikely to have many terms satisfied simultaneously.

Lemma 17 Let $f = T_1 \vee \dots \vee T_t$ be a read-once DNF formula of size t such that $\Pr[f] < 1 - \epsilon$. Then the probability over the uniform distribution on $\{0, 1\}^n$ that some set of $j > e \ln 1/\epsilon$ terms is satisfied is at most $\left(\frac{e \ln 1/\epsilon}{j}\right)^j$.

Proof: For any assignment x to the variables of f , let $y_f(x)$ be the number terms satisfied in f . By linearity of expectation, we have that $\mathbf{E}_x[y_f(x)] = \sum_{i=1}^t \Pr[T_i = 1]$. Note that $\Pr[\neg f] = \prod_{i=1}^t (1 - \Pr[T_i])$, which is maximized when each $\Pr[T_i] = \mathbf{E}[y_f]/t$, hence $\Pr[\neg f] \leq (1 - \mathbf{E}[y_f]/t)^t \leq e^{-\mathbf{E}[y_f]}$. Thus we may assume that $\mathbf{E}[y_f] \leq \ln 1/\epsilon$, otherwise $\Pr[f] \geq 1 - \epsilon$.

Assuming $\mathbf{E}[y_f] \leq \ln 1/\epsilon$, we now bound the probability that some set of $j > e \ln 1/\epsilon$ terms of f is satisfied. Since all the terms are disjoint, this probability is $\sum_{S \subseteq [t], |S|=j} \prod_{i \in S} \Pr[T_i]$, and the arithmetic-geometric mean inequality gives that this is maximized when every $\Pr[T_i] = \mathbf{E}[y_f]/t$. Then the probability of satisfying some set of j terms is at most:

$$\binom{t}{j} \left(\frac{\ln 1/\epsilon}{t}\right)^j \leq \binom{et}{j} \left(\frac{\ln 1/\epsilon}{t}\right)^j = \left(\frac{e \ln 1/\epsilon}{j}\right)^j,$$

which concludes the proof of the lemma. ■

The following lemma shows that we can set d to be fairly small, $\Theta(\log 1/\epsilon)$, and the polynomial $p_{f,d}$ will be a good approximation for any DNF formula f , as long as f is unlikely to have many terms satisfied simultaneously.

Lemma 18 Let f be any t -term DNF formula, and let $d = \lceil 4e^3 \ln 1/\epsilon \rceil$. If

$$\Pr[y_f(x) = j] \leq \left(\frac{e \ln 1/\epsilon}{j}\right)^j$$

for every $d \leq j \leq t$, then the polynomial $p_{f,d}$ satisfies $\mathbf{E}[(f - p_{f,d})^2] \leq \epsilon$.

Proof: We condition on the values of $y_f(x)$, controlling the magnitude of $p_{f,d}$ by the unlikelihood of y_f being large. By Fact 15, $p_{f,d}(x)$ will output 0 if x does not satisfy f , $p_{f,d}(x)$ will output 1 if $y_f(x) \in [d]$, and $|p_{f,d}(x)| < \binom{y_f}{d}$ for $y_f(x) \in [t] \setminus [d]$. Hence:

$$\begin{aligned} \|f - p_{f,d}\|^2 &< \sum_{j=d+1}^t \binom{j}{d}^2 \left(\frac{e \ln 1/\epsilon}{j}\right)^j \\ &< \sum_{j=d+1}^t 2^{2j} \left(\frac{e \ln 1/\epsilon}{4e^3 \ln 1/\epsilon}\right)^j \\ &< \epsilon \sum_{j=d+1}^t \frac{1}{e^j} < \epsilon. \end{aligned}$$

Combining Lemmas 16, 17, and 18 gives us Mansour's conjecture for read-once DNF formulas.

Theorem 19 Let f be any read-once DNF formula with t terms. Then there is a polynomial $p_{f,d}$ with $\|p_{f,d}\|_1 \leq t^{O(\log 1/\epsilon)}$ and $\mathbf{E}[(f - p_{f,d})^2] \leq \epsilon$ for all $\epsilon > 0$.

4 Mansour's Conjecture for Random DNF Formulas

In this section, we establish various properties of random DNF formulas and use these properties to show that for almost all f , Mansour's conjecture holds. Roughly speaking, we will show that a random DNF formula behaves like a read-once DNF formula, in that any "large" set of terms is unlikely to be satisfied by a random assignment. This notion is formalized in Lemma 22. For such DNF formulas, we may use the construction from Section 3 to obtain a good approximating polynomial for f with small spectral norm (Theorem 24).

Throughout the rest of this section, we assume that $t(n) = n^{O(1)}$. For brevity we write t for $t(n)$. Let \mathcal{D}_n^t be the probability distribution over t -term DNF formulas induced by the following process: each term is independently and uniformly chosen at random from all $\binom{n}{\log t}$ possible terms of size exactly $\log t$ over $\{x_1, \dots, x_n\}$. For convenience, we assume that $\log t$ is an integer throughout our discussion, although the general case is easily handled by taking terms of length $\lfloor \log t \rfloor$. If the terms are not of size $\Theta(\log n)$, then the DNF will be biased, and thus be easy to learn. We refer the reader to Jackson and Servedio (2005) for a full discussion of the model.

If t grows very slowly relative to n , say $t = n^{o(1)}$, then with high probability $(1 - n^{-\Omega(1)})$ a random f drawn from \mathcal{D}_n^t will be a read-once DNF formula, in which case the results of Section 3.1 hold. Therefore, throughout the rest of this section we will assume that t is in fact $n^{\Theta(1)}$.

To prove Lemma 22, we require two lemmas, which are inspired by the results of (Jackson & Servedio, 2005) and (Jackson et al., 2008). Lemma 20 shows that with high probability the terms of a random DNF formula are close to being disjoint, and thus cover close to $j \log t$ variables.

Lemma 20 *With probability at least $1 - t^j e^{j \log t} (j \log t)^{\log t} / n^{\log t}$ over the random draw of f from \mathcal{D}_n^t , at least $j \log t - (\log t)/4$ variables occur in every set of j distinct terms of f . The failure probability is at most $1/n^{\Omega(\log t)}$ for any $j < c \log n$, for some constant c .*

Proof: Let $k := \log t$. Fix a set of j terms, and let $v \leq jk$ be the number of distinct variables (negated or not) that occur in these terms. We will bound the probability that $v > w := jk - k/4$. Consider any particular fixed set of w variables. The probability that none of the j terms include any variable outside of the w variables is precisely $\left(\binom{w}{k} / \binom{n}{k}\right)^j$. Thus, the probability that $v \leq w$ is by the union bound:

$$\binom{n}{w} \left(\frac{\binom{w}{k}}{\binom{n}{k}} \right)^j < \left(\frac{en}{w} \right)^w \left(\frac{w}{n} \right)^{jk} = \frac{e^{jk - k/4} (jk - k/4)^{k/4}}{n^{k/4}} < \frac{e^{jk} (jk)^{k/4}}{n^{k/4}}.$$

Taking a union bound over all (at most t^j) sets, we have that with the correct probability every set of j terms contains at least w distinct variables. \blacksquare

We will use the method of bounded differences (a.k.a., McDiarmid's inequality) to prove Lemma 22.

Proposition 21 (McDiarmid's inequality) *Let X_1, \dots, X_m be independent random variables taking values in a set \mathcal{X} , and let $f : \mathcal{X}^m \rightarrow \mathbb{R}$ be such that for all $i \in [m]$, $|f(a) - f(a')| \leq d_i$, whenever $a, a' \in \mathcal{X}^m$ differ in just the i th coordinate. Then for all $\tau > 0$,*

$$\Pr[f > \mathbf{E}f + \tau] \leq \exp\left(-\frac{2\tau^2}{\sum_i d_i^2}\right) \text{ and } \Pr[f < \mathbf{E}f - \tau] \leq \exp\left(-\frac{2\tau^2}{\sum_i d_i^2}\right).$$

The following lemma shows that with high probability over the choice of random DNF formula, the probability that exactly j terms are satisfied is close to that for the "tribes" function: $\binom{t}{j} t^{-j} (1 - 1/t)^{t-j}$.

Lemma 22 *There exists a constant c such that for any $j < c \log n$, with probability at least $1 - 1/n^{\Omega(\log t)}$ over the random draw of f from \mathcal{D}_n^t , the probability over the uniform distribution on $\{0, 1\}^n$ that an input satisfies exactly j distinct terms of f is at most $2 \binom{t}{j} t^{-j} (1 - 1/t)^{t-j}$.*

Proof: Let $f = T_1 \vee \dots \vee T_t$, and let $\beta := t^{-j} (1 - 1/t)^{t-j}$. Fix any $J \subset [t]$ of size j , and let U_J be the probability over $x \in \{0, 1\}^n$ that the terms T_i for $i \in J$ are satisfied and no other terms are satisfied. We will show that $U_J < 2\beta$ with high probability; a union bound over all possible sets J of size j in $[t]$ gives that $U_J \leq 2\beta$ for every J with high probability. Finally, a union bound over all $\binom{t}{j}$ possible sets of j terms (where the probability is taken over x) proves the lemma.

Without loss of generality, we may assume that $J = [j]$. For any fixed x , we have:

$$\Pr_{f \in \mathcal{D}_n^t} [x \text{ satisfies exactly the terms in } J] = \beta,$$

and thus by linearity of expectation, we have $\mathbf{E}_{f \in \mathcal{D}_n^t} [U_J] = \beta$. Now we show that with high probability that the deviation of U_J from its expected value is low.

Applying Lemma 20, we may assume that the terms T_1, \dots, T_j contain at least $j \log t - (\log t)/4$ many variables, and that $J \cup T_i$ for all $i = j+1, \dots, t$ includes at least $(j+1) \log t - (\log t)/4$ many unique variables, while increasing the failure probability by only $1/n^{\Omega(\log t)}$. Note that conditioning on this event can change the value of U_J by at most $1/n^{\Omega(\log t)} < \frac{1}{2}\beta$, so under this conditioning we have $\mathbf{E}[P_j] \geq \frac{1}{2}\beta$. Conditioning on this event, fix the terms T_1, \dots, T_j . Then the terms T_{j+1}, \dots, T_t are chosen uniformly and independently from the set of all terms T of length $\log t$ such that the union of the variables in J and T includes at least $(j+1) \log t - (\log t)/4$ unique variables. Call this set \mathcal{X} .

We now use McDiarmid's inequality where the random variables are the terms T_{j+1}, \dots, T_t randomly selected from \mathcal{X} , letting $g(T_{j+1}, \dots, T_t) = U_J$ and $g(T_{j+1}, \dots, T_{i-1}, T'_i, T_{i+1}, \dots, T_t) = U'_J$ for all $i = j+1, \dots, t$. We claim that:

$$|U_J - U'_J| \leq d_i := \frac{t^{1/4}}{t^{j+1}}.$$

This is because U'_J can only be larger than U_J by assignments which satisfy T_1, \dots, T_j and T_i . Similarly, U'_J can only be smaller than U_J by assignments which satisfy T_1, \dots, T_j and T'_i . Since T_i and T'_i come from \mathcal{X} , we know that at least $(j+1)t - (\log t)/4$ variables must be satisfied.

Thus we may apply McDiarmid's inequality with $\tau = \frac{3}{2}\beta$, which gives that $\Pr_f[U_J > 2\beta]$ is at most

$$\exp\left(\frac{-2\frac{9}{4}\beta^2}{t^{3/2}/t^{2j+2}}\right) \leq \exp\left(\frac{-9\sqrt{t}(1-1/t)^{2(t-j)}}{2}\right).$$

Combining the failure probabilities over all the $\binom{t}{j}$ possible sets, we get that with probability at least

$$\binom{t}{j} \left(\frac{1}{n^{\Omega(\log t)}} + e^{-9\sqrt{t}(1-1/t)^{2(t-j)}/2} \right) = \frac{1}{n^{\Omega(\log t)}},$$

over the random draw of f from \mathcal{D}_n^t , U_J for all $J \subseteq [t]$ of size j is at most 2β . Thus, the probability that a random input satisfies exactly some j distinct terms of f is at most $2\binom{t}{j}\beta$. ■

Using these properties of random DNF formulas we can now show a lemma analogous to Lemma 18 for random DNF formulas.

Lemma 23 *Let f be any DNF formula with $t = n^{O(1)}$ terms, and let $\epsilon > 0$ which satisfies $1/\epsilon = o(\log \log n)$. Then set $d = \lceil 4e^3 \ln 1/\epsilon \rceil$ and $\ell = c \log n$, where c is the constant in Lemma 22. If*

$$\Pr[y_f(x) = j] \leq \left(\frac{e \ln 1/\epsilon}{j}\right)^j$$

for every $d \leq j \leq \ell$, then the polynomial $p_{f,d}$ satisfies $\mathbf{E}[(f - p_{f,d})^2] \leq \epsilon$.

Proof: We condition on the values of $y_f(x)$, controlling the magnitude of $p_{f,d}$ by the unlikelihood of y_f being large. By Fact 15, $p_{f,d}(x)$ will output 0 if x does not satisfy f , $p_{f,d}(x)$ will output 1 if $y_f(x) \in [d]$, and $|p_{f,d}(x)| < \binom{y_f}{d}$ for $y_f(x) \in [t] \setminus [d]$. Hence:

$$\begin{aligned} \|f - p_{f,d}\|^2 &< \sum_{j=d+1}^{\ell-1} \binom{j}{d}^2 \left(\frac{e \ln 1/\epsilon}{j}\right)^j + \binom{t}{d}^2 \cdot \Pr[y_f \geq \ell] \\ &< \sum_{j=d+1}^{\ell-1} 2^{2j} \left(\frac{e \ln 1/\epsilon}{4e^3 \ln 1/\epsilon}\right)^j + n^{-\Omega(\log \log n)} \\ &< \epsilon \sum_{j=d+1}^{\ell-1} \frac{1}{e^j} + n^{-\Omega(\log \log n)} < \epsilon. \end{aligned}$$

We can now show that Mansour's conjecture (Mansour, 1994) is true with high probability over the choice of f from \mathcal{D}_n^t .

Theorem 24 *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a $t = n^{\Theta(1)}$ -term DNF formula where each term is chosen independently from the set of all terms of length $\log t$. Then with probability at least $1 - n^{-\Omega(\log t)}$ over the choice of f , there exists a polynomial p with $\|p\|_1 \leq t^{O(\log 1/\epsilon)}$ such that $\mathbf{E}[(p(x) - f(x))^2] \leq \epsilon$.*

Proof: Let $d := \lceil 4e^3 \ln(1/\epsilon) \rceil$ and $p_{f,d}$ be as defined in Section 3. Lemma 16 tells us that $\|p_{f,d}\|_1 \leq t^{O(\log 1/\epsilon)}$. We show that with probability at least $1 - n^{-\Omega(\log t)}$ over the random draw of f from \mathcal{D}_n^t , $p_{f,d}$ will be a good approximator for f . This follows by Lemma 22; with probability at least $1 - (c \log(n) - d - 1)/n^{\Omega(\log t)} = 1 - n^{-\Omega(\log t)}$, we have $\Pr[y = j]$ for all $d < j \leq c \log(n)$. Thus for such f Lemma 18 tells us that $\mathbf{E}[(f - p_{f,d})^2] \leq \epsilon$. \blacksquare

5 Mansour's Conjecture for Read- k DNF Formulas

In this section, we give an ϵ -approximating polynomial for any read- k DNF formula and show that its spectral norm is at most $t^{O(2^{4k} \log 1/\epsilon)}$. This implies that Mansour's conjecture holds for all read- k DNF formulas where k is any constant.

Read- k DNF formulas may not satisfy the conditions of Lemma 18, so we must change our approach. Instead of using $\sum_{i=1}^t T_i$ inside our univariate polynomial, we use a different sum, which is based on a construction from (Razborov, 2008) for representing any DNF formula. We modify this representation to exploit the fact that for read- k DNF formulas, the variables in a term can not share variables with too many other terms. Unlike for read-once DNF formulas, it is not clear that the number of terms satisfied in a read- k DNF formula will be extremely concentrated on a small range. We show how to modify our construction so that a concentration result does hold.

Let $f = T_1 \vee \dots \vee T_t$ be any t -term read- k DNF formula, and let $|T_i|$ denote the number of variables in the term T_i . We assume that the terms are ordered from longest to shortest, i.e., $|T_j| \geq |T_i|$ for all $j \leq i$. For any term T_i of f , let ϕ_i be the DNF formula consisting of those terms (at least as large as T_i) in T_1, \dots, T_{i-1} that overlap with T_i , i.e.,

$$\phi_i := \bigvee_{j \in \mathcal{C}_i} T_j, \text{ for } \mathcal{C}_i = \{j < i \mid T_j \cap T_i \neq \emptyset\}.$$

We define $A_i := T_i \wedge \neg \phi_i$ and $z_f := \sum_{i=1}^t A_i$. The function $z_f : \{0, 1\}^n \rightarrow \{0, \dots, t\}$ outputs the number of disjoint terms of f satisfied by x (greedily starting from T_1). Note that if f is a read-once DNF formula, then $z_f = y_f$.

Observe that each A_i can be represented by the polynomial $T_i \cdot \prod_{j \in \mathcal{C}_i} (1 - T_j)$ (and so z_f can be represented by a polynomial), and that $\|(1 - T_j)\|_1 \leq 2$ for all j . As f is a read- k DNF formula, each ϕ_i has at most $k|T_i|$ terms, and A_i has small spectral norm:

Fact 25 *Let $f = T_1 \vee \dots \vee T_t$ be a t -term read- k DNF formula. Then each A_i has a polynomial representation, and $\|A_i\|_1 \leq 2^{k|T_i|}$.*

As we did in Section 3, we can construct a polynomial $q_{f,d} : \{0, 1\}^n \rightarrow \mathbb{R}$ defined as $q_{f,d} := P_d \circ z_f$ for any t -term read- k DNF formula f . The following lemma shows that $q_{f,d}$ has small spectral norm.

Lemma 26 *Let f be a t -term read- k DNF formula with terms of length at most w . Then $\|q_{f,d}\|_1 \leq 2^{O(d(\log t + kw))}$.*

Proof: By Fact 15, P_d is a degree- d univariate polynomial with d terms and coefficients of magnitude at most d . We can view the polynomial $q_{f,d}$ as the polynomial $P'_d(A_1, \dots, A_t) := P_d(A_1 + \dots + A_t)$ over variables $A_i \in \{0, 1\}$. Expanding out (but not recombining) P'_d gives us at most dt^d monomials of degree d (over variables A_i) with coefficients of magnitude at most d .

We can now apply Facts 25 and 9 to bound the spectral norm of $q_{f,d}$. Since P'_d has at most dt^d monomials each of degree d (over A_i), and each A_i satisfies $\|A_i\|_1 \leq 2^{kw}$, we have that $\|q_{f,d}\|_1 \leq 2^{dkw} dt^d = 2^{O(d(\log t + kw))}$. \blacksquare

We will show that Mansour's conjecture holds for read- k DNF formulas by showing that $z_f = \sum_{i=1}^t A_i$ behaves much like $y_f = \sum_{i=1}^t T_i$ would if f were a read-once DNF formula, and thus we can use our polynomial P_d (Equation 1) to approximate f .

One crucial property of our construction is that only disjoint sets of terms can contribute to z_f .

Claim 27 *Let $T_1 \vee \dots \vee T_t$ be a t -term DNF formula. Then for any $S \subseteq [t]$, $\Pr[\wedge_{i \in S} A_i] \leq \prod_{i \in S} \Pr[T_i]$.*

Proof: If there is a pair $j, k \in S$ such that $T_j \cap T_k \neq \emptyset$ for some $j < k$, then ϕ_k contains T_j and both $T_j \wedge \neg \phi_j$ and $T_k \wedge \neg \phi_k$ cannot be satisfied simultaneously, so $\Pr[\wedge_{i \in S} A_i] = 0$. If no such pair exists, then all the terms indexed by S are disjoint. Thus,

$$\Pr[\wedge_{i \in S} A_i] \leq \Pr[\wedge_{i \in S} T_i] = \prod_{i \in S} \Pr[T_i],$$

as was to be shown. ■

The following lemma was communicated to us by Omid Etesami and James Cook (Etesami & Cook, 2010).

Lemma 28 *Let $f = T_1 \vee \dots \vee T_t$ be a t -term read- k DNF formula, and let $f' = T'_1 \vee \dots \vee T'_t$ be the monotone formula obtained from f by replacing all the negative literals by their positive counterparts. Then $\Pr[f'] \leq \Pr[f]$.*

Proof: For each $0 \leq i \leq n$, define $f^{(i)}$ as the DNF formula obtained from f when replacing each occurrence of $\neg x_j$ by x_j for all $1 \leq j \leq i$. In particular, $f^{(0)} = f$ and $f^{(n)} = f'$. Let $f^{(i-1)} = (g_{x_i} \wedge x_i) \vee (g_{\neg x_i} \wedge \neg x_i) \vee g_\emptyset$ where $g_{x_i} \wedge x_i$ is the OR of all terms from $f^{(i-1)}$ that have the literal x_i , $g_{\neg x_i} \wedge \neg x_i$ is the OR of all terms that have the literal $\neg x_i$, and g_\emptyset is the OR of all terms that neither contain x_i nor contain $\neg x_i$. Note that $f^{(i)} = ((g_{x_i} \vee g_{\neg x_i}) \wedge x_i) \vee g_\emptyset$. Thus

$$\Pr[f^{(i-1)}] = \frac{1}{2} \Pr[g_{x_i} \wedge \neg g_\emptyset] + \frac{1}{2} \Pr[g_{\neg x_i} \wedge \neg g_\emptyset] + \Pr[g_\emptyset],$$

and

$$\Pr[f^{(i)}] = \frac{1}{2} \Pr[(g_{x_i} \vee g_{\neg x_i}) \wedge \neg g_\emptyset] + \Pr[g_\emptyset].$$

A union bound on the events $(g_{x_i} \wedge \neg g_\emptyset)$ and $(g_{\neg x_i} \wedge \neg g_\emptyset)$ tells us that $\Pr[f^{(i-1)}] \geq \Pr[f^{(i)}]$, and thus $\Pr[f^{(0)}] \geq \Pr[f^{(n)}]$. ■

As in the read-once case, we will prove that for any read- k DNF formula f , if $\sum_{i=1}^t \Pr[T_i]$ is large then f is biased towards one (Lemma 30). To do so we will prove this for monotone read- k DNF formulas and then use Lemma 28 to obtain the general case. Before we prove Lemma 30 we need the following claim, which tells us that for a read- k monotone DNF formula, the probability of satisfying A_i compared to that of satisfying T_i is only smaller by a constant (for constant k).

Claim 29 *Let $T_1 \vee \dots \vee T_t$ be a t -term monotone read- k DNF formula. Then $2^{-4k} \Pr[T_i] \leq \Pr[A_i]$.*

Proof: Let I be the set of indices of the terms in ϕ_i . For each $T_j \in \phi_i$, let T'_j be T_j with all the variables of T_i set to 1, and let $\phi'_i = \vee_{\{j: T_j \in \phi_i\}} T'_j$. (For example, if $T_i = x_1 x_2 x_3$ and $T_j = x_2 x_4 x_5$ is a term of ϕ_i , then ϕ'_i contains the term $T'_j = x_4 x_5$.) Observe that $\Pr[A_i] = \Pr[T_i \wedge \neg \phi'_i] = \Pr[T_i] \Pr[\neg \phi'_i]$. Thus it suffices to show that $\Pr[\neg \phi'_i] \geq 2^{-4k}$.

Let a_j be the number of variables in $T_j \cap T_i$. By the definition of ϕ_i , $1 \leq a_j \leq |T_i| - 1$, and note that $\Pr[T'_j] = 2^{a_j - |T_j|}$. Applying the Four Functions Theorem (Theorem 6), we obtain:

$$\Pr[\neg \phi'_i] \geq \prod_{j \in I} \Pr[\neg T'_j] = \prod_{j \in I} (1 - 2^{a_j - |T_j|}) \geq \prod_{j \in I} (1 - 2^{a_j - |T_i|}).$$

We partition I into two sets: $J = \{j : a_j \leq |T_i|/2\}$ and $J' = \{j : a_j > |T_i|/2\}$. (Assume that $|T_i| \geq 4$ or else we are done, because there can be at most $4k$ terms.) As ϕ_i is a read- k DNF formula, we have that $\sum_{j \in I} a_j \leq k|T_i|$, and thus $|J'| \leq 2k$, and $|J| \leq k|T_i|$.

We will lower bound the products over each set of indices separately. For those terms in J , we have that $\Pr[T'_j] \leq 2^{-|T_i|/2}$, hence

$$\prod_{j \in J} (1 - \Pr[T'_j]) \geq \prod_{j \in J} (1 - 2^{-|T_i|/2}) \geq (1 - 2^{-|T_i|/2})^{k|T_i|} \geq 2^{-2k}.$$

For those terms T_j , $j \in J'$ (which share many variables with T_i), we use the facts that each $\Pr[T'_j] \leq 1/2$ and that there are at most $2k$ such terms, so that

$$\prod_{j \in J'} (1 - \Pr[T'_j]) \geq 2^{-2k}.$$

Taking the product over the set $J \cup J'$ completes the proof of the claim. ■

Finally, we will prove that for any read- k DNF formula f , if $\sum_{i=1}^t \Pr[T_i]$ is large then f is biased towards one. Using Lemma 30 with Claim 27, we can prove a lemma analogous to Lemma 17 by a case analysis of $\sum_{i=1}^t \Pr[T_i]$; either it is large and f must be biased toward one, or it is small so z_f is usually small.

Lemma 30 *Let f be a t -term read- k DNF formula. Then,*

$$\sum_{i=1}^t \Pr[T_i] \leq 2^{4k} \ln \left(\frac{1}{\Pr[\neg f]} \right).$$

Proof: First, let us consider the case when f is monotone. Let ρ_i be those terms among T_1, \dots, T_{i-1} that are not present in ϕ_i . We can upper-bound $\Pr[\neg f]$ by:

$$\begin{aligned} \Pr[\neg f] &= \prod_{i=1}^t (1 - \Pr[T_i \mid \neg\phi_i \wedge \neg\rho_i]) \\ &\leq \prod_{i=1}^t (1 - \Pr[T_i \wedge \neg\phi_i \mid \neg\rho_i]) = \prod_{i=1}^t (1 - \Pr[T_i \mid \neg\rho_i] \Pr[\neg\phi_i \mid T_i \wedge \neg\rho_i]) \\ &\leq \prod_{i=1}^t (1 - \Pr[T_i] \Pr[\neg\phi_i \mid T_i]) = \prod_{i=1}^t (1 - \Pr[A_i]). \end{aligned}$$

The first inequality comes from $\Pr[A \mid B \wedge C] \geq \Pr[A \wedge B \mid C]$ for any A, B , and C . The last inequality holds because $\Pr[T_i \mid \neg\rho_i] = \Pr[T_i]$ (by the mutual independence of T_i and ρ_i) and $\Pr[\neg\phi_i \mid T_i] \leq \Pr[\neg\phi_i \mid T_i \wedge \neg\rho_i]$. The last fact may be obtained by applying the Four Functions Theorem to $\neg\phi_i$ and $\neg\rho_i$ under the product distribution induced by setting all the variables of T_i to be true.

We apply Claim 29 to obtain $\Pr[\neg f] \leq \prod_{i=1}^t (1 - \Pr[T_i] 2^{-4k})$, and the arithmetic-geometric mean inequality shows that our upper-bound on $\Pr[\neg f]$ is maximized when all the $\Pr[T_i]$ are equal, hence:

$$\Pr[\neg f] \leq \left(1 - 2^{-4k} \frac{\sum_{i=1}^t \Pr[T_i]}{t} \right)^t \leq \exp \left(-2^{-4k} \sum_{i=1}^t \Pr[T_i] \right).$$

Solving for $\sum_{i=1}^t \Pr[T_i]$ yields the lemma.

Now let f be a non-monotone DNF formula, and let f' be the monotonized version of f . Then by Lemma 28 we have:

$$\sum_{i=1}^t \Pr[T_i] = \sum_{i=1}^t \Pr[T'_i] \leq 2^{4k} \ln \left(\frac{1}{\Pr[\neg f']} \right) \leq 2^{4k} \ln \left(\frac{1}{\Pr[\neg f]} \right),$$

as was to be shown. ■

Lemma 31 *Let $f = T_1 \vee \dots \vee T_t$ be a read- k DNF formula of size t such that $\Pr[f] < 1 - \epsilon$. Then the probability over the uniform distribution on $\{0, 1\}^n$ that $z_f \geq j$ (for any $j > 2^{4k} e \ln(1/\epsilon)$) is at most $\left(\frac{2^{4k} e \ln(1/\epsilon)}{j} \right)^j$.*

Proof: By Lemma 30, $T_A := \sum_{i=1}^t \Pr[T_i] < 2^{4k} \ln(1/\epsilon)$. The probability that some set of j A_i 's is satisfied is at most $\sum_{S \subseteq [t], |S|=j} \Pr[\wedge_{i \in S} A_i]$. Applying Claim 27, we have:

$$\sum_{S \subseteq [t], |S|=j} \Pr[\wedge_{i \in S} A_i] \leq \sum_{S \subseteq [t], |S|=j} \prod_{i \in S} \Pr[T_i].$$

The arithmetic-geometric mean inequality shows that this quantity is maximized when all $\Pr[T_i]$ are equal, hence:

$$\sum_{S \subseteq [t], |S|=j} \prod_{i \in S} \Pr[T_i] \leq \binom{t}{j} \left(\frac{T_A}{t} \right)^j \leq \left(\frac{\epsilon T_A}{j} \right)^j \leq \left(\frac{2^{4k} e \ln 1/\epsilon}{j} \right)^j$$

We can now show that Mansour's conjecture holds for read- k DNF formulas with any constant k .

Theorem 32 *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be any read- k DNF formula with t terms. Then there is a polynomial $q_{f,d}$ with $\|q_{f,d}\|_1 = t^{O(2^{4k} \log 1/\epsilon)}$ and $\mathbf{E}[(f - q_{f,d})^2] \leq \epsilon$ for all $\epsilon > 0$.*

Proof: If $\Pr[f = 1] > 1 - \epsilon$, the constant 1 is a suitable polynomial. Let g be the DNF formula f after dropping terms of length greater than $w := \log(2t/\epsilon)$. (This only changes the probability by $\epsilon/2$.) Let $d := \lceil 4e^3 2^{4k} \ln(2/\epsilon) \rceil$ and $q_{g,d}$ be as defined at the beginning of Section 5. Lemma 26 tells us that $\|q_{g,d}\|_1 \leq t^{O(2^{4k} \log 1/\epsilon)}$, and Lemma 31 combined with Lemma 18 tells us that $\mathbf{E}[(g - q_{g,d})^2] \leq \epsilon/2$. ■

6 Pseudorandomness

De et al. (2009) recently improved long-standing pseudorandom generators against DNF formulas.

Definition 33 A probability distribution X over $\{0, 1\}^n$ ϵ -fools a real function $f : \{0, 1\}^n \rightarrow \mathbb{R}$ if

$$|\mathbf{E}[f(X)] - \mathbf{E}[f(U_n)]| \leq \epsilon.$$

If \mathcal{C} is a class of functions, then we say that X ϵ -fools \mathcal{C} if X ϵ -fools every function $f \in \mathcal{C}$.

We say a probability distribution X over $\{0, 1\}^n$ is ϵ -biased if it ϵ -fools the character function χ_S for every $S \subseteq [n]$.

De et al. (2009) observed that the result of Bazzi (2007) implied a pseudorandom generator that ϵ -fools t -term DNF formulas over n variables with seed length $O(\log n \cdot \log^2(t/\beta))$, which already improves the long-standing upper bound of $O(\log^4(tn/\epsilon))$ of Luby et al. (1993). They go on to show a pseudorandom generator with seed length $O(\log n + \log^2(t/\epsilon) \log \log(t/\epsilon))$.

They prove that a sufficient condition for a function f to be ϵ -fooled by an ϵ -biased distribution is that the function be “sandwiched” between two bounded real-valued functions whose Fourier transform has small ℓ_1 norm:

Lemma 34 (Sandwich Bound (De et al., 2009)) Suppose $f, f_\ell, f_u : \{0, 1\}^n \rightarrow \mathbb{R}$ are three functions such that for every $x \in \{0, 1\}^n$, $f_\ell(x) \leq f(x) \leq f_u(x)$, $\mathbf{E}[f_u(U_n)] - \mathbf{E}[f(U_n)] \leq \epsilon$, and $\mathbf{E}[f(U_n)] - \mathbf{E}[f_\ell(U_n)] \leq \epsilon$. Let $L = \max(\|f_\ell\|_1^{\neq 0}, \|f_u\|_1^{\neq 0})$. Then any β -biased probability distribution $(\epsilon + \beta L)$ -fools f .

Naor and Naor (1993) prove that an ϵ -biased distribution over n bits can be sampled using a seed of $O(\log(n/\epsilon))$ bits. Using our construction from Section 4, we show that random DNF formulas are ϵ -fooled by a pseudorandom generator with seed length $O(\log n + \log(t) \log(1/\epsilon))$:

Theorem 35 Let $f = T_1 \vee \dots \vee T_t$ be a random DNF formula chosen from \mathcal{D}_n^t for $t = n^{\Theta(1)}$. For $1 \leq d \leq t$, with probability $1 - 1/n^{\Omega(\log t)}$ over the choice of f , β -biased distributions $O(2^{-\Omega(d)} + \beta t^d)$ -fool f . In particular, we can ϵ -fool most $f \in \mathcal{D}_n^t$ by a $t^{-O(\log(1/\epsilon))}$ -biased distribution.

Proof: Let d^+ be the first odd integer greater than d , and let d^- be the first even integer greater than d . Let $f_u = p_{f, d^+}$ and $f_\ell = p_{f, d^-}$ (where $p_{f, d}$ is defined as in Section 3). By Lemma 16, the ℓ_1 -norms of f_u and f_ℓ are $t^{O(d)}$. By Fact 15, we know that $P_{d^+}(y) = \binom{y-1}{d} + 1 > 1$ and $P_{d^-}(y) = -\binom{y-1}{d} + 1 \leq 0$ for $y \in [t] \setminus [d]$, hence:

$$\mathbf{E}[f_u(U_n)] - \mathbf{E}[f(U_n)] = \sum_{j=d+1}^t \left(\binom{j-1}{d} + 1 - 1 \right) \Pr[y_f = j],$$

which with probability $1 - 1/n^{\Omega(\log t)}$ over the choice of f is at most $2^{-\Omega(d)}$ by the analysis in Lemma 18. The same analysis applies for f_ℓ , thus applying Lemma 34 gives us the theorem. \blacksquare

De et al. (2009) match our bound for random DNF formulas for the special case of read-once DNF formulas. Using our construction from Section 5 and a similar proof as the one above, we can show that monotone read- k formulas are ϵ -fooled by a pseudorandom generator with seed length $O(\log n + \log(t) \log(1/\epsilon))$.

Theorem 36 Let $f = T_1 \vee \dots \vee T_t$ be a read- k DNF formula for constant k . For $1 \leq d \leq t$, β -biased distributions $O(2^{-\Omega(d)} + \beta t^d)$ -fool f . In particular, we can ϵ -fool read- k DNF formulas by a $t^{-O(\log(1/\epsilon))}$ -biased distribution.

Acknowledgments. Thanks to Sasha Sherstov for important contributions at an early stage of this work. We would also like to thank Omid Etesami for pointing out some errors in a previous version of this work, including a crucial flaw in the proof of the read- k case. We also thank him for pointing out to us that our proof for the monotone read- k case extends to the non-monotone case (through Lemma 28). Lemma 28 is due to Omid and James Cook.

References

- Alon, N., & Spencer, J. H. (2000). *The probabilistic method*. Hoboken, NJ: Wiley-Interscience. 2nd edition edition.
- Bazzi, L. (2007). Polylogarithmic independence can fool DNF formulas. *Proc. 48th IEEE Symposium on Foundations of Computer Science (FOCS)* (pp. 63–73).
- De, A., Etesami, O., Trevisan, L., & Tulsiani, M. (2009). *Improved pseudorandom generators for depth 2 circuits* (Technical Report 141). Electronic Colloquium on Computational Complexity (ECCC).
- Etesami, O., & Cook, J. (2010). personal communication.
- Gopalan, P., Kalai, A., & Klivans, A. R. (2008a). A query algorithm for agnostically learning DNF? *21st Annual Conference on Learning Theory - COLT 2008, Helsinki, Finland, July 9-12, 2008* (pp. 515–516). Omnipress.
- Gopalan, P., Kalai, A. T., & Klivans, A. R. (2008b). Agnostically learning decision trees. *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008* (pp. 527–536). ACM.
- Graham, R. L., Knuth, D. E., & Patashnik, O. (1994). *Concrete mathematics: A foundation for computer science*. Addison-Wesley.
- Hancock, T., & Mansour, Y. (1991). Learning monotone k - μ DNF formulas on product distributions. *Proc. of the 4th Annual Conference on Computational Learning Theory (COLT)* (pp. 179–183).
- Håstad, J. (1986). *Computational limitations for small depth circuits*. MIT Press.
- Jackson, J. C. (1997). An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55, 414–440.
- Jackson, J. C., Lee, H. K., Servedio, R. A., & Wan, A. (2008). Learning random monotone DNF. *12th Intl. Workshop on Randomization and Computation (RANDOM)* (pp. 483–497). Springer-Verlag.
- Jackson, J. C., & Servedio, R. A. (2005). On learning random DNF formulas under the uniform distribution. *9th Intl. Workshop on Randomization and Computation (RANDOM)* (pp. 342–353). Springer-Verlag.
- Kalai, A., Klivans, A., Mansour, Y., & Servedio, R. (2008). Agnostically learning halfspaces. *SIAM Journal on Computing*, 37, 1777–1805.
- Kleitman, D. J. (1966). Families of non-disjoint subsets. *Journal of Combinatorial Theory*, 1, 153–155.
- Kushilevitz, E., & Mansour, Y. (1993). Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22, 1331–1348. Prelim. ver. in *Proc. of STOC'91*.
- Linial, N., Mansour, Y., & Nisan, N. (1993). Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40, 607–620.
- Luby, M., Velickovic, B., & Wigderson, A. (1993). Deterministic approximate counting of depth-2 circuits. *ISTCS 1993* (pp. 18–24).
- Mansour, Y. (1994). *Learning Boolean functions via the Fourier transform*, 391–424. Kluwer Academic Publishers.
- Mansour, Y. (1995). An $O(n^{\log \log n})$ learning algorithm for DNF under the uniform distribution. *Journal of Computer and System Sciences*, 50, 543–550. Prelim. ver. in *Proc. of COLT'92*.
- Naor, J., & Naor, M. (1993). Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22, 838–856.
- Razborov, A. (2008). *A simple proof of Bazzi's theorem* (Technical Report 81). Electronic Colloquium on Computational Complexity (ECCC).
- Sellie, L. (2008). Learning random monotone DNF under the uniform distribution. *Proc. of the 21th Annual Conference on Computational Learning Theory (COLT)* (pp. 181–192).
- Sellie, L. (2009). Exact learning of random DNF over the uniform distribution. *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)* (pp. 45–54).

Deterministic Sparse Fourier Approximation via Fooling Arithmetic Progressions

Adi Akavia*

The Weizmann Institute of Science, Rehovot, Israel.

adi.akavia@weizmann.ac.il

Abstract

A significant Fourier transform (SFT) algorithm, given a threshold τ and oracle access to a function f , outputs (the frequencies and approximate values of) all the τ -significant Fourier coefficients of f , i.e., the Fourier coefficients whose magnitude exceeds $\tau\|f\|_2^2$. In this paper we present the first *deterministic* SFT algorithm for functions f over \mathbb{Z}_N which is: (1) Local, i.e., its running time is polynomial in $\log N$, $1/\tau$ and $L_1(\hat{f})$ (the L_1 norm of f 's Fourier transform). (2) Robust to random noise. This strictly extends the class of compressible/Fourier sparse functions over \mathbb{Z}_N efficiently handled by prior deterministic algorithms. As a corollary we obtain deterministic and robust algorithms for sparse Fourier approximation, compressed sensing and sketching.

As a central tool, we prove that there are:

1. Explicit sets A of size $\text{poly}((\ln N)^d, 1/\varepsilon)$ with ε -discrepancy in all rank d Bohr sets in \mathbb{Z}_N . This extends the Razborov-Szemerédi-Wigderson result on ε -discrepancy in arithmetic progressions to Bohr sets, which are their higher rank analogue.
2. Explicit sets A_P of size $\text{poly}(\ln N, 1/\varepsilon)$ that ε -approximate the uniform distribution over a given arithmetic progression P in \mathbb{Z}_N , in the sense that $|\mathbb{E}_{x \in A} \chi(x) - \mathbb{E}_{x \in P} \chi(x)| < \varepsilon$ for all linear tests χ in \mathbb{Z}_N . This extends results on small biased sets, which are sets approximating the uniform distribution over the entire domain, to sets approximating uniform distributions over (arbitrary size) arithmetic progressions.

These results may be of independent interest.

1 Introduction

Computing the Fourier transform is a basic building block used in numerous applications. Its complexity is well understood: Quasi-linear running time $O(N \log N)$ for N the input size is achieved by the Fast Fourier Transform (FFT) algorithm [CT65], and believed to be optimal. For data intensive applications, however, achieving a *sub-linear* running time is desired. In general, this is infeasible, because the input and output are already of size N . Nevertheless, in settings where the input is given via *oracle access*, and it suffices to output only the few “*significant*” Fourier coefficients, sub-linear algorithms do exist.

We say that a Fourier coefficient is τ -significant if its magnitude is at least a τ -fraction (say, 1%) of the sum of squared Fourier coefficients. A significant Fourier transform (SFT) algorithm is an algorithm that, given a significance threshold τ and oracle access to a function f , outputs all the τ -significant Fourier coefficients of f (i.e., their frequencies and approximate values). The complexity of such algorithms is measured primarily in terms of $1/\tau$ and the size N of (the truth table of) f .

Randomized SFT algorithms achieving complexity polynomial in $\log N$ and $1/\tau$ for functions over any finite abelian group were developed in a sequence of works [GL89, KM93, Man95, GGI⁺02, AGS03, GMS05, Aka09].

Deterministic SFT algorithms were given for restricted functions:

- Functions over the *boolean hypercube* $\{0, 1\}^n$ in Kushilevitz-Mansour’s (KM) algorithm [KM93].

*This research was supported in part by NSF grant CCF-0514167, by NSF grant CCF-0832797, and by Israel Science Foundation 700/08.

- *Compressible* or *Fourier sparse* functions over \mathbb{Z}_N in Iwen’s algorithms [Iwe07, Iwe08, IS08]. (A function is **compressible** if its Fourier coefficients decay as fast as the series $c(1/i)^p$ for absolute constants $c > 0$ and $p > 1$. A function is **Fourier sparse** if it has at most $\text{poly}(\log N)$ non-zero Fourier coefficients.)

The KM algorithm [KM93] is given as an extra input an upper bound t on the sum of (absolute values of) Fourier coefficients of the input function f , $L_1(\hat{f}) \stackrel{\text{def}}{=} \sum_{\alpha} |\hat{f}(\alpha)|$. Its running time is polynomial in $\log N$, $1/\tau$ and t . We say that a deterministic SFT algorithm *achieves the KM benchmark* if its complexity is polynomial in $\log N$, $1/\tau$ and t .

1.1 Main Result: Deterministic & Robust SFT Algorithm

In this paper we present a deterministic SFT algorithm achieving the KM benchmark for all functions f over \mathbb{Z}_N . Furthermore, our SFT algorithm is *robust* to random noise. That is, the algorithm succeeds even if the oracle to f is noisy in the sense that on queries x the oracle returns the value $f'(x) = f(x) + \eta(x)$ for $\eta: \mathbb{Z}_N \rightarrow \mathbb{C}$ an ε -**random noise**, i.e., values $\eta(x)$ are drawn independently at random from distributions D_x of expected absolute value at most $\mathbb{E}_{\eta(x) \sim D_x} [|\eta(x)|] \leq \varepsilon$.

Theorem 1 *There is a deterministic algorithm such that:*

- Given N , τ , t , and oracle access to a function $f: \mathbb{Z}_N \rightarrow \mathbb{C}$ s.t. $L_1(\hat{f}) \leq t$, the algorithm outputs all the τ -significant Fourier coefficients of f .
- Given N , τ , t , and oracle access is to a function $f': \mathbb{Z}_N \rightarrow \mathbb{C}$, where $f' = f + \eta$ s.t. $L_1(\hat{f}) \leq t$ and η is a $\tau/3$ -random noise, the algorithm outputs all the τ -significant Fourier coefficients of f (with probability at least $1 - 1/N^{\Theta(1)}$ over the random noise η).

The running time and query complexity are polynomial in $\log N$, $1/\tau$ and t .

Remarks. (i) The KM benchmark is matched by taking $t = L_1(\hat{f})$. (ii) We stress that the complexity of our algorithm depends on the bound t on $L_1(\hat{f})$, and not on a bound on $L_1(\hat{f}')$. This is crucial, because even if $L_1(\hat{f}) \leq t$ is small, typically $L_1(\hat{f}') \approx \sqrt{N}$ is very large.

Our algorithm is better than the prior deterministic SFT algorithms for functions over \mathbb{Z}_N in:

1. Achieving the KM benchmark. In particular, our algorithm efficiently (i.e., in time polynomial in $\log N$) handles a much wider class of functions than handled by prior works: all functions f s.t. $L_1(\hat{f}) \leq \text{poly}(\log N)$, instead of only the compressible/Fourier sparse functions.¹ Handling this wider class of functions is motivated by functions arising in applications, e.g., threshold functions $f_{\theta}(x) = 1$ iff $x \leq \theta$ and 0 otherwise.
2. Achieving robustness to random noise. In contrast, in other deterministic algorithms, noisy functions $f' = f + \eta$ are out of the scope of functions handled efficiently, because typically f' is not compressible/Fourier sparse (even if f were).²

Robustness to noise is motivated for example by measurement noise in signal processing applications.

1.2 New Tools: Fooling Bohr Sets and Arithmetic Progressions

As a central ingredient for our deterministic SFT algorithm, we prove that there exists explicit constructions of: (1) Sets with small discrepancy on all rank d Bohr sets; and (2) Sets that ε -approximate the uniform distribution on a given arithmetic progression (definitions follow). These results may be of independent interest.

¹In the context of SFT algorithms, compressible functions f are a strict subclass of the functions with poly-logarithmic $L_1(\hat{f})$. This is because without loss of generality we may assume that f is normalized to have (approximately) unit energy (as significance is determined by *ratios* of Fourier coefficient magnitude to total energy), and for normalized f , compressibility implies that $L_1(\hat{f}) = O(1)$.

²A few remarks. (i) Having some restriction on the class of functions efficiently handled by deterministic algorithms is unavoidable (because two input functions f, g may be identical on the small set of entries read by the deterministic algorithm, while differing widely on their Fourier transform, implying the algorithm fails on at least one out of f, g). The class of functions handled by our algorithm is wide enough to include typical noise. (ii) Our analysis extends to show that KM’s deterministic algorithm for functions over $\{0, 1\}^n$ is also robust to random noise.

1.2.1 Definitions

For sets A, S in a group G , we denote by U_S the uniform distribution over S , and say that:

- A has ε -discrepancy on S in G if the intersection $|A \cap S|$ is roughly as expected if A were random:

$$D_{A,G}(S) \stackrel{\text{def}}{=} \left| \frac{|A \cap S|}{|A|} - \frac{|S|}{|G|} \right| < \varepsilon.$$

For a family \mathcal{S} of sets, A has ε -discrepancy on \mathcal{S} , if $D_{A,G}(\mathcal{S}) \stackrel{\text{def}}{=} \max_{S \in \mathcal{S}} D_{A,G}(S) < \varepsilon$.

- A ε -approximates U_S in G if for all linear tests $\chi: G \rightarrow \mathbb{C}$ in G , the expected outcome $\chi(x)$ over uniform x in A is ε -close to its expected outcome over uniform x in S :

$$\left| \mathbb{E}_{x \in A} \chi(x) - \mathbb{E}_{x \in S} \chi(x) \right| < \varepsilon.$$

We focus on $G = \mathbb{Z}_N$, where linear tests are the functions $\chi_\alpha(x) \stackrel{\text{def}}{=} e^{2\pi i \alpha x / N}$ indexed by $\alpha \in \mathbb{Z}_N$.

- A is **explicit** if there is a deterministic algorithm that, given G (by its generators and their orders) and ε , outputs A in time polynomial in $|A|$. We usually focus on explicit sets A of size polynomial in $\log |G|$ and $1/\varepsilon$.

We are particularly interested in sets S that are either arithmetic progressions or Bohr sets (which are the higher rank analogue of arithmetic progressions used in many additive combinatorics works [TV06]):

- **Arithmetic progressions** in \mathbb{Z}_N are sets $P_{\alpha,I} \stackrel{\text{def}}{=} \{x \cdot \alpha \bmod N \mid x \in I\}$ for $\alpha \in \mathbb{Z}_N$ a multiplier and $I = [a..b]$ an interval (i.e., the set of integers in $[a, b]$) with endpoints $0 \leq a \leq b < N$.
- **Rank d Bohr sets** in \mathbb{Z}_N are sets $B_{\{\alpha_i, I_i\}_{i=1}^d} \stackrel{\text{def}}{=} \{x \in \mathbb{Z}_N \mid \alpha_i \cdot x \bmod N \in I_i \forall i = 1, \dots, d\}$ for $\alpha_i \in \mathbb{Z}_N$ multipliers and $I_i = [a_i..b_i]$ intervals with endpoints $0 \leq a_i \leq b_i < N$. Denote by $\mathcal{B}_{N,d}$ the set of all rank d Bohr sets in \mathbb{Z}_N .

1.2.2 Our Results

We show that there exists (1) explicit sets with small discrepancy on all rank d Bohr sets, and (2) explicit sets approximating the uniform distribution on a given arithmetic progression:

Theorem 2 1. For any N, ε, d , there is an explicit set $A \subseteq \mathbb{Z}_N$ of size polynomial in $1/\varepsilon$ and $(\ln N)^d$ with ε -discrepancy on $\mathcal{B}_{N,d}$.

2. For any N, ε , and an arithmetic progression P in \mathbb{Z}_N , there is an explicit set $A \subseteq \mathbb{Z}_N$ of size polynomial in $1/\varepsilon$ and $\ln |P|$ that ε -approximates the distribution U_P .

Remarks and comparison to prior works.

1. Our proof is by reduction to explicit constructions of small biased sets. The exact size of our sets A depends on the small biased set we use. For example, using the ε' -biased set of size $O((\log N)^2/\varepsilon')$ of [Kat89] results in sets A of sizes $O((\log N)^{2+d}/\varepsilon)$ and $O((\log N)^4/\varepsilon^3)$ in Theorem 2 Parts 1 and 2 respectively.³
2. For $d = 1$, our proof of Theorem 2 Part 1 gives a new –and much simpler– proof for the Razborov-Szemerédi-Wigderson [RSW93] result on ε -discrepancy on arithmetic progressions. Our result (even when restricted to $d = 1$) is better than the latter in: (i) Achieving ε -discrepancy on all arithmetic progressions $P_{\alpha,I}$, in contrast to only $P_{\alpha,I}$ s.t. α is co-prime to N . (ii) Achieving a better set size whenever $\varepsilon < 1/(\log N)^{1/8}$ (as for $d = 1$ our set size is $\Theta((\log N)^3/\varepsilon)$ compared with $\Theta((\log N)^2/\varepsilon^9)$ in [RSW93]).
3. Theorem 2 Part 2 can be viewed as a generalization of small biased sets in \mathbb{Z}_N : Small biased sets are sets approximating the uniform distribution over the entire domain, that is, the arithmetic progression $P_{\alpha,I}$ for $\alpha = 1$ and $I = [0..N - 1]$, whereas our result addresses arbitrary arithmetic progressions P .

³The size quoted here for the ε' -biased set of [Kat89] is taken from the accounts of [AIK⁺90] on [Kat89]. We remark that using the (much simpler) ε' -biased set of size $O((\log N)^2/\varepsilon^3)$ of [AIK⁺90] results in sets A of sizes $O((\log N)^{2+d}/\varepsilon^3)$ and $O((\log N)^4/\varepsilon^9)$ in Theorem 2 Parts 1 and 2 respectively.

1.3 Paper Organization

In the rest of this paper we present: An overview of our proof (Sect. 2); Preliminaries (Sect. 3); Our explicit constructions (Sect. 4); Our deterministic SFT algorithm (Sect. 5); Concluding remarks (Sect. 6).

2 Proof Overview

Our starting point is the randomized SFT algorithm of [Aka09]. Randomness there is employed solely for constructing a set $S = S_{N,\tau,t} \subseteq \mathbb{Z}_N$ of queries to the oracle to the input function f , such that (with high probability) S is good according to Definition 3 below. Their analysis shows that: (i) If S is good, then for all functions f over \mathbb{Z}_N s.t. $L_1(\widehat{f}) \leq t$, their algorithm finds the τ -significant Fourier coefficients of f (even in the presence of noise) while querying f only on the entries in S . (ii) With high probability, their S is good.

Definition 3 (Good queries [Aka09]) A set $S = S_{N,\tau,t} \subseteq \mathbb{Z}_N$ is (N, τ, t) -good (good, in short) if $S = \bigcup_{\ell=0}^{\lfloor \log N \rfloor} (A - B_\ell)$ s.t. for $\varepsilon = \Theta(\tau/(t^2 \log N))$ sufficiently small:

- A is an ε -biased set in \mathbb{Z}_N
- For each ℓ , B_ℓ ε -approximates the distribution $U_{[0..2^\ell-1]}$ in \mathbb{Z}_N
- The sizes $|A|$ and $|B_1|, \dots, |B_{\lfloor \log N \rfloor}|$ are polynomial in $\log N$, $1/\tau$ and t

Remark. Exact setting of ε may vary depending on the desired tradeoff: We take $\varepsilon = \tau/(3t^2 \ln N)$ when there is no noise, and $\varepsilon = \tau/(49t^2 \ln N)$ to tolerate up to $\tau/3$ -random noise.

In this work we obtain a deterministic (and robust) SFT algorithm by replacing the randomized construction of sets S in [Aka09] with an explicit (i.e., efficient and deterministic) construction.

The heart of our construction is a novel analysis, where, for any arithmetic progression P (say, $[0..2^\ell]$), we reduce the problem of finding explicit sets approximating the distribution U_P in \mathbb{Z}_N to the problem of finding explicit sets with small bias in \mathbb{Z}_M for $M = |P|$.⁴ We then obtain a good set S by utilizing known constructions of explicit small-biased sets in \mathbb{Z}_M [Kat89, AIK⁺90, RSW93].

Our reduction is composed of the three parts detailed in the theorem below.

Theorem 4 (Our reduction) For any positive integers $d, M < N$, a positive real ε , and $A \subseteq I = [0..M-1]$,

1. If A is $\varepsilon/(4 \ln M)^d$ -biased in \mathbb{Z}_M , then A has ε -discrepancy on all rank d Bohr sets in \mathbb{Z}_M .
2. If A has $\varepsilon^3/(128\pi^2)$ -discrepancy on all rank 2 Bohr sets in \mathbb{Z}_M , then A ε -approximates U_I in \mathbb{Z}_N .
3. For every $\alpha, s \in \mathbb{Z}_N$, if A ε -approximates U_I in \mathbb{Z}_N , then $\alpha(A + s)$ ε -approximates $U_{P_{\alpha, I+s}}$ in \mathbb{Z}_N .

Remark. The converse of Theorem 4 Part 1 is known (and simple to prove): If A has ε -discrepancy on all arithmetic progressions in \mathbb{Z}_N , then A is $2\pi\varepsilon$ -biased in \mathbb{Z}_N (see [RSW93], Proposition 4.1).

Proving part 3 of our reduction is straightforward, whereas Parts 1-2 require more insight (details follow).

Theorem 4, Part 1. We relate having small bias in \mathbb{Z}_M to having small discrepancy on all rank d Bohr sets in \mathbb{Z}_M as follows. First we upper bound the discrepancy of A on any set R (say, a rank d Bohr set) in \mathbb{Z}_M by $D_{A, \mathbb{Z}_N}(R) \leq \varepsilon' \cdot L_1(\widehat{R})$ for ε' the bias of A in \mathbb{Z}_M and for $L_1(\widehat{R}) \stackrel{\text{def}}{=} \sum_{\alpha \in \mathbb{Z}_M} |\widehat{R}(\alpha)|$ the L_1 -norm of the Fourier transform of (the characteristic function of) R . Next we apply Fourier analysis and some elementary number theory to show that for any rank d Bohr set R in \mathbb{Z}_M , $L_1(\widehat{R}) \leq (4 \ln M)^d$. We conclude that if A is $\varepsilon' = \varepsilon/(4 \ln M)^d$ -biased in \mathbb{Z}_M , then A has ε -discrepancy on all rank d Bohr sets in \mathbb{Z}_M .

Theorem 4, Part 2. We relate approximating U_I in \mathbb{Z}_N to having small discrepancy on rank 2 Bohr sets in \mathbb{Z}_M as follows.

First, we identify each element $\alpha \in \mathbb{Z}_N$ with the pair (q_α, r_α) of its quotient and remainder in the division-with-remainder by (the typically, non-integer value) N/M . We then rewrite each linear test $\chi_\alpha(x) = e(\alpha x/N)$ in \mathbb{Z}_N as: $\chi_\alpha(x) = e\left(\left(\left(\frac{q_\alpha x}{M}\right)_1 + \left(\frac{r_\alpha x}{N}\right)_1\right)_1\right)$ (where for any real number r , $(r)_1$ denotes its non-integer part, i.e., its remainder modulo 1; and $e(r) = e^{2\pi i r}$).

Second, we embed the sets $\left\{\left(\frac{q_\alpha x}{M}\right)_1\right\}_{x \in I}$ and $\left\{\left(\frac{r_\alpha x}{N}\right)_1\right\}_{x \in I}$ into \mathbb{Z}_M , and use this embedding to show that if A has ε' -discrepancy on all rank 2 Bohr sets in \mathbb{Z}_M , then the joint distribution of pairs $\left(\left(\frac{q_\alpha x}{M}\right)_1, \left(\frac{r_\alpha x}{N}\right)_1\right)$

⁴We remark that the connection between approximating U_P in \mathbb{Z}_N and having small bias in $\mathbb{Z}_{|P|}$ may seem surprising. For example, achieving the former requires satisfying N linear tests modulo N , whereas achieving the latter requires satisfying only $|P|$ linear tests and these tests are modulo $|P|$ (where $|P| < N$ is arbitrary).

over uniform x in A is “close” to their distribution over uniform x in I ; where closeness is in the sense that for every two length $\rho = \varepsilon/8\pi$ intervals $J_1, J_2 \subseteq [0, 1]$,

$$\left| \Pr_{x \in A} \left[\left(\frac{q_\alpha x}{M} \right)_1 \in J_1 \ \& \ \left(\frac{r_\alpha x}{N} \right)_1 \in J_2 \right] - \Pr_{x \in I} \left[\left(\frac{q_\alpha x}{M} \right)_1 \in J_1 \ \& \ \left(\frac{r_\alpha x}{N} \right)_1 \in J_2 \right] \right| < \varepsilon'$$

Finally, we prove that ε' -closeness of these two joint distributions implies $((\varepsilon'/\rho^2) + 4\pi\rho)$ -closeness of the expected value of $\chi_\alpha(x) = e\left(\left(\left(\frac{q_\alpha x}{M}\right)_1 + \left(\frac{r_\alpha x}{N}\right)_1\right)_1\right)$ over uniform x in A and over uniform x in I . Assigning $\varepsilon' = \varepsilon^3/(128\pi^2)$, we conclude that A ε -approximates U_I in \mathbb{Z}_N .

3 Preliminaries

In this section we summarize some preliminary terminology, notations and facts.

Notations. Let \mathbb{Z} and \mathbb{C} denote the integer and complex numbers respectively. Let \mathbb{Z}_N and \mathbb{Z}_N^* denote the additive and the multiplicative groups of integers modulo N . We identify the elements of \mathbb{Z}_N with integers in $0, \dots, N-1$, and denote $\text{abs}(\alpha) = \min\{\alpha, N-\alpha\}$ for all $\alpha \in \mathbb{Z}_N$. We denote by $[a..b]$ the set of integers in the closed interval $[a, b]$. For any element $a \in \mathbb{Z}_N$ and sets $S, S' \subseteq \mathbb{Z}_N$, denote $aS = \{as\}_{s \in S}$ and $S - S' = \{s - s'\}_{s \in S, s' \in S'}$. For any real number r , denote $e(r) = e^{2\pi ir}$, and denote by $(r)_1$ the remainder of r in division by 1.

3.1 Significant Fourier Transform Coefficients

We give definitions and properties for normed spaces and Fourier transform.

Inner product, norms, convolution. The *inner product* of complex valued functions f, g over a domain G is $\langle f, g \rangle \stackrel{\text{def}}{=} \frac{1}{|G|} \sum_{x \in G} f(x)\overline{g(x)}$. Denote the ℓ_2 -norm of f by $\|f\|_2 \stackrel{\text{def}}{=} \sqrt{\langle f, f \rangle}$, and its L_1 -norm by $L_1(f) \stackrel{\text{def}}{=} \sum_{x \in G} |f(x)|$. The *convolution* of f and g is the function $f * g: G \rightarrow \mathbb{C}$ defined by $f * g(x) \stackrel{\text{def}}{=} \frac{1}{|G|} \sum_{y \in G} f(y)g(x-y)$.

Characters and Fourier transform. The *characters* of a finite abelian groups G are all the homomorphisms $\chi: G \rightarrow \mathbb{C}$ from G into the complex unit sphere. Denote by \widehat{G} the set of characters of G . The *Fourier transform* of a complex valued function f over G is the function $\widehat{f}: \widehat{G} \rightarrow \mathbb{C}$ defined by $\widehat{f}(\chi) \stackrel{\text{def}}{=} \langle f, \chi \rangle$. A character χ is *trivial* if $\chi(x) = 1$ for all x .

In particular, the characters of \mathbb{Z}_N are the functions $\chi_\alpha: \mathbb{Z}_N \rightarrow \mathbb{C}$, $\alpha \in \mathbb{Z}_N$, defined by $\chi_\alpha(x) \stackrel{\text{def}}{=} e^{2\pi i \alpha x/N}$. Abusing notation, we view \widehat{f} as a function over G , defined by $\widehat{f}(\alpha) \stackrel{\text{def}}{=} \langle f, \chi_\alpha \rangle$.

Significant Fourier coefficients. For any $\alpha \in \mathbb{Z}_N$ and $\tau \in [0, 1]$, we say that α is a τ -*significant* Fourier coefficient iff $|\widehat{f}(\alpha)|^2 \geq \tau \|f\|_2^2$. Denote by $\text{Heavy}_\tau(f)$ the set of all τ -significant Fourier coefficients of f .

Useful Fourier transform properties. For every positive integer N , functions $f, g: \mathbb{Z}_N \rightarrow \mathbb{C}$, and elements $s \in \mathbb{Z}_N$ and $t \in \mathbb{Z}_N^*$, the following holds (where subtraction, multiplication and inverse operations are modulo N): *Parseval Identity*: $\frac{1}{N} \sum_{x \in \mathbb{Z}_N} |f(x)|^2 = \sum_{\alpha \in \mathbb{Z}_N} |\widehat{f}(\alpha)|^2$. *Convolution Theorem*: $\widehat{(f * g)}(\alpha) = \widehat{f}(\alpha) \cdot \widehat{g}(\alpha)$ and similarly $\frac{1}{N} \widehat{f \cdot g}(\alpha) = (\widehat{f} * \widehat{g})(\alpha)$. *Phase Shift*: If $g = f \cdot \chi_{-s}$, then $\widehat{g}(\alpha) = \widehat{f}(\alpha - s)$ for all α . *Scaling*: If $g(x) = f(tx) \ \forall x$, then $\widehat{g}(\alpha) = \widehat{f}(\alpha \cdot t^{-1})$ for all α . Finally, for all integers α , $\frac{1}{N} \sum_{x \in \mathbb{Z}_N} e(\alpha x/N) = 1$ iff $\alpha = 0 \pmod N$ and it is 0 otherwise.

3.2 Small Biased Sets

Let G be a finite abelian group and $A \subseteq G$. Denote $\text{bias}_A(\chi) \stackrel{\text{def}}{=} \frac{1}{|A|} \sum_{a \in A} \chi(a)$. A is ε -*biased* in G if for all non-trivial characters χ of G , $|\text{bias}_A(\chi)| < \varepsilon$.

Fact 1 (ε -biased sets in \mathbb{Z}_N [AIK⁺90, RSW93, Kat89]) For any integer $N > 0$ and real $\varepsilon > 0$, there exists an explicit set A of size polynomial in $\log N$ and $1/\varepsilon$ which is ε -biased in \mathbb{Z}_N .

For the sake of completeness we specify the details of one of the small biased sets constructed in [AIK⁺90] (chosen due to its simplicity): For each N, ε , let $\text{AIKPS}(N, \varepsilon)$ denote the ε -biased set in \mathbb{Z}_N of [AIK⁺90] defined by:

$$\text{AIKPS}(N, \varepsilon) \stackrel{\text{def}}{=} \{sp^{-1} \pmod N \mid s \in S, p \in P, p \nmid s\} \tag{1}$$

where, for $\varepsilon' = -(\log \varepsilon)/(\log \log N)$, $P = \{p \mid p \text{ is prime, } (\log N)^{1+\varepsilon'}/2 < p \leq (\log N)^{1+\varepsilon'}\}$ and $S = [1..(\log N)^{1+2\varepsilon'}]$.

4 Our Results on Explicit Constructions

We present our results on explicit sets with small discrepancy on Bohr sets (Sect. 4.1), and explicit sets approximating distributions uniform over arithmetic progressions (Sect. 4.2). See definitions in Sect. 1.2.1.

4.1 Small Discrepancy in Bohr Set

We show that there are explicit small sets A with small discrepancy on all rank d Bohr sets in \mathbb{Z}_N .

Theorem 2 Part 1 (Small discrepancy on Bohr sets). For any integers $N, d > 0$ and real $\varepsilon > 0$, there is an explicit set $A \subseteq \mathbb{Z}_N$ of size polynomial in $1/\varepsilon$ and $(\ln N)^d$ s.t. $D_{A, \mathbb{Z}_N}(\mathcal{B}_{N,d}) < \varepsilon$.

Proof: Fix N, d, ε . Denote $\varepsilon' = \varepsilon/(4 \ln N)^d$. Let A be any explicit ε' -biased set A in \mathbb{Z}_N of size polynomial in $\log N$ and $1/\varepsilon'$ (such sets exists by Fact 1, Sect. 3.2). By Lemma 5, for any set $B \subseteq \mathbb{Z}_N$,

$$D_{A, \mathbb{Z}_N}(B) < \varepsilon' L_1(\widehat{B})$$

for $L_1(\widehat{B}) \stackrel{\text{def}}{=} \sum_{\alpha \in \mathbb{Z}_N} |\widehat{B}(\alpha)|$ the L_1 -norm of the Fourier transform of the characteristic function $B(x) = 1$ iff $x \in B$ and 0 otherwise. By Lemma 6, for any rank d Bohr set B ,

$$L_1(\widehat{B}) \leq (4 \ln N)^d.$$

We conclude that $D_{A, \mathbb{Z}_N}(B) < \varepsilon' \cdot (4 \ln N)^d = \varepsilon$. ■

For any subsets A, B of a finite abelian group G , we bound the discrepancy of A on B in G by the product of the bias of A in G and the L_1 -norm of the Fourier transform of B .

Lemma 5 For any finite abelian group G , sets $A, B \subseteq G$ and a real number $\varepsilon > 0$, if A is ε -biased in G , then $D_{A,G}(B) < \varepsilon L_1(\widehat{B})$. Furthermore, this bound is tight.

Proof: We show that $D_{A,G}(B) < \varepsilon L_1(\widehat{B})$. Recall that $D_{A,G}(B) = \left| \frac{|A \cap B|}{|A|} - \frac{|B|}{|G|} \right|$. Replacing B with its characteristic function, we get that $D_{A,G}(B) = \left| \frac{1}{|A|} \sum_{a \in A} B(a) - \frac{1}{|G|} \sum_{a \in G} B(a) \right|$. Replacing the characteristic function of B with its Fourier representation $B(a) = \sum_{\chi} \widehat{B}(\chi) \chi(a)$ and rearranging the summation we get that $D_{A,G}(B) = \left| \sum_{\chi} \widehat{B}(\chi) \left(\frac{1}{|A|} \sum_{a \in A} \chi(a) - \frac{1}{|G|} \sum_{a \in G} \chi(a) \right) \right|$ which is in turn equal to

$$D_{A,G}(B) = \left| \sum_{\text{non-trivial } \chi} \widehat{B}(\chi) \cdot \text{bias}_A(\chi) \right| \tag{2}$$

(since for the trivial χ , the difference in the parenthesis is zero, and for all non-trivial χ , $\frac{1}{|G|} \sum_{a \in G} \chi(a) = 0$). Using the triangle inequality and bounding the bias of A by its upper bound ε , we conclude that

$$D_{A,G}(B) \leq \max_{\text{non trivial } \chi} |\text{bias}_A(\chi)| \sum_{\chi} |\widehat{B}(\chi)| < \varepsilon L_1(\widehat{B}).$$

We prove the bound is tight. Let $G = \{0, 1\}^n$. Given any ε -biased set $A \subseteq \{0, 1\}^n$ and $\alpha \in \{0, 1\}^n$ s.t. $\text{bias}_A(\chi_{\alpha}) = \varepsilon$, define $B = \{x \mid x \cdot \alpha = 0\}$ ($x \cdot \alpha$ is the dot product). Then $D_{A,G}(B) = \varepsilon L_1(\widehat{B})$, because $D_{A,G}(B) = \left| \sum_{\alpha \neq 0} \widehat{B}(\chi_{\alpha}) \text{bias}_A(\chi_{\alpha}) \right|$ (by Eq. 2) and $\widehat{B}(\chi_{\beta}) \neq 0$ iff $\beta = \alpha$ (by Fourier analysis of B). ■

We bound the L_1 -norm of the Fourier transform of rank d Bohr sets B .

Lemma 6 For any positive integers N, d , and B the characteristic function of a rank d Bohr set in \mathbb{Z}_N , $L_1(\widehat{B}) \leq (4 \ln N)^d$.

Proof: Fix $B_{\{\alpha_i, I_i\}_{i=1}^d}$ a rank d Bohr set in \mathbb{Z}_N . Observe that $B_{\{\alpha_i, I_i\}_{i=1}^d} = \bigcap_{i=1}^d B_{\{\alpha_i, I_i\}}$ is the intersection of the d rank 1 Bohr sets $B_{\{\alpha_i, I_i\}}$. Denote by B and B_1, \dots, B_d the characteristic functions of $B_{\{\alpha_i, I_i\}_{i=1}^d}$ and $B_{\{\alpha_1, I_1\}}, \dots, B_{\{\alpha_d, I_d\}}$ respectively. Then $B = \prod_{i=1}^d B_i$. By Proposition 2, the above implies that $L_1(\widehat{B}) \leq \prod_{i=1}^d L_1(\widehat{B}_i)$. By Lemma 7, $L_1(\widehat{B}_i) \leq 4 \ln N$ for any rank 1 Bohr set B_i . We conclude that $L_1(\widehat{B}) \leq (4 \ln N)^d$. ■

We bound the L_1 -norm of the Fourier transform of a product of functions.

Proposition 2 Let $f, g: G \rightarrow \mathbb{C}$, then $L_1(\widehat{f \cdot g}) \leq L_1(\widehat{f}) \cdot L_1(\widehat{g})$.

Proof: By the convolution theorem, $\widehat{f \cdot g}(\chi) = N(\widehat{f * \widehat{g}})(\chi)$. Thus, $L_1(\widehat{f \cdot g}) = \sum_{\chi} |N(\widehat{f * \widehat{g}})(\chi)|$. By definition of the convolution operator, the latter is equal to the sum $\sum_{\chi} \left| \sum_{\psi} \widehat{f}(\psi) \cdot \widehat{g}(\chi \cdot \psi^{-1}) \right|$ over χ, ψ characters of the group G , which is upper bounded by $\sum_{\psi} |\widehat{f}(\psi)| \sum_{\chi} |\widehat{g}(\chi \cdot \psi^{-1})| = L_1(\widehat{f}) \cdot L_1(\widehat{g})$. ■

We bound the L_1 -norm of the Fourier transform of rank 1 Bohr sets.

Lemma 7 For any positive integer N , and B the characteristic function of any rank 1 Bohr set in \mathbb{Z}_N , $L_1(\widehat{B}) \leq 4 \ln N$.

Remark. The bound is tight up to constants, that is, there are Bohr sets B s.t. $L_1(\widehat{B}) = \Omega(\ln N)$.

Proof: Fix a rank 1 Bohr set in \mathbb{Z}_N , $B = B_{\alpha, I}$, for $\alpha \in \mathbb{Z}_N$ and $I = [s..t] \subseteq \mathbb{Z}_N$. Denote by $g \stackrel{def}{=} \gcd(N, \alpha)$ the greatest common divisor of N and α , and let $\beta \stackrel{def}{=} \alpha/g$. We prove that $L_1(\widehat{B}) \leq 4 \ln N$.

For $g = 1$ the proof is simple: Since α is co-prime to N , then $\mathbb{Z}_N = \{\alpha^{-1}x \bmod N\}_{x \in \mathbb{Z}_N}$, implying that B is equal to the arithmetic progression $\{(\alpha^{-1}x \bmod N) \in \mathbb{Z}_N \mid \alpha(\alpha^{-1}x) \in I\} = \alpha^{-1}I$. By the scaling property of the Fourier transform, for α^{-1} co-prime to N , $L_1(\widehat{\alpha^{-1}I}) = L_1(\widehat{I})$. Thus, by Proposition 3, $L_1(\widehat{B}) \leq 4 \ln N$.

For $g > 1$ the proof is more involved, details are given in Sect. 4.1.1. ■

We bound the L_1 -norm of the Fourier transform of an interval I .

Proposition 3 Let I be the characteristic function of an interval in \mathbb{Z}_N . Then $L_1(\widehat{I}) < 4 \ln N$.

Proof: By [AGS03], the Fourier coefficients of any length at most $N/2$ interval I' are upper bounded by $|\widehat{I'}(\alpha)| \leq 1/\text{abs}(\alpha)$ for $\text{abs}(\alpha) = \min\{\alpha, N - \alpha\}$, and $|\widehat{I}(0)| \leq 1$. For longer intervals I , we write I as the sum of two intervals each of length at most $N/2$. By linearity of the Fourier transform, the Fourier coefficients of I are the sum of the Fourier coefficients of these intervals, and are therefore bounded by $\widehat{I}(\alpha) \leq 2/\text{abs}(\alpha)$. We conclude that $L_1(\widehat{I}) \leq 1 + 2 \sum_{\alpha=1}^{N/2} (2/\text{abs}(\alpha)) < 4 \ln N$ (where for the last inequality we use the bound $\sum_{i=1}^n (1/i) \leq 1 + \ln n$ on harmonic numbers). ■

4.1.1 Proof of Lemma 7

We prove Lemma 7 for the case of rank 1 Bohr sets in \mathbb{Z}_N with a multiplier α that is not co-prime to N .

Proof of Lemma 7. Fix a rank 1 Bohr set $B \stackrel{def}{=} \{x \in \mathbb{Z}_N \mid (\alpha x \bmod N) \in I\}$ in \mathbb{Z}_N for $\alpha \in \mathbb{Z}_N$ and $I = [s..t] \subseteq \mathbb{Z}_N$. Denote by $g \stackrel{def}{=} \gcd(N, \alpha)$ the greatest common divisor of N and α , and let $\beta \stackrel{def}{=} \alpha/g$. We prove that $L_1(\widehat{B}) \leq 4 \ln N$, focusing on the case that $g > 1$.

By Claim 8, for $J \stackrel{def}{=} } \left[\left(\frac{s}{g} \right) \right] .. \left[\left(\frac{t}{g} \right) \right]$ we can rewrite B as

$$B = \left\{ \beta^{-1} \left(i \frac{N}{g} + x_0 \right) \in \mathbb{Z}_N \mid x_0 \in J, i \in [0..g-1] \right\} \quad (3)$$

for β^{-1} the inverse of β modulo N . Denote

$$\widehat{J}_{N/g}(\alpha) \stackrel{def}{=} \frac{1}{N/g} \sum_{x \in J} e(\alpha x / (N/g)).$$

By Claim 9, for every index $\gamma \in \mathbb{Z}_N$, the γ -Fourier coefficient of B is

$$\widehat{B}(\gamma) = \begin{cases} 0 & g \nmid \gamma \\ \widehat{J}_{N/g}(\gamma \beta^{-1} / g) & g \mid \gamma \end{cases}$$

Thus,

$$L_1(\widehat{B}) = \sum_{\gamma' \in \mathbb{Z}_{N/g}} \widehat{J}_{N/g}((\gamma' g) \beta^{-1} / g) = \sum_{\gamma' \in \mathbb{Z}_{N/g}} \left| \widehat{J}_{N/g}(\gamma' \beta^{-1}) \right| = \sum_{\gamma' \in \mathbb{Z}_{N/g}} \left| \widehat{J}_{N/g}(\gamma') \right|$$

where the last equality holds because β^{-1} is co-prime to N/g (because it is co-prime to N , and N/g is a divisor of N). By the definition of $\widehat{J_{N/g}}, \sum_{\gamma' \in \mathbb{Z}_{N/g}} |\widehat{J_{N/g}}(\gamma')|$ is equal to $L_1(\widehat{J})$ where the Fourier transform of J is computed with respect to the characters of $\mathbb{Z}_{N/g}$. By Proposition 3, the latter is upper bounded by $4 \ln(N/g) \leq 4 \ln N$. We conclude that $L_1(\widehat{B}) \leq 4 \ln N$. ■

Claim 8 $B = \left\{ \beta^{-1}(i \frac{N}{g} + x_0) \in \mathbb{Z}_N \mid x_0 \in J, i \in [0..g-1] \right\}$.

Proof: Since $\{\beta^{-1}x\}_{x \in \mathbb{Z}_N} = \mathbb{Z}_N$ (as by the maximality of $g, \gcd(N, \beta) = 1$), then B is equal to the set $\{\beta^{-1}x \in \mathbb{Z}_N \mid (\alpha(\beta^{-1}x) \bmod N) \in I\}$. Since $\alpha(\beta^{-1}x) = (g\beta)(\beta^{-1}x) = gx$ we get that

$$B = \{ \beta^{-1}x \in \mathbb{Z}_N \mid (gx \bmod N) \in I \}.$$

Thus, by Proposition 4, for $J = \left[\left[\left(\frac{s}{g} \right) \right] .. \left[\left(\frac{t}{g} \right) \right] \right]$,

$$B = \left\{ \beta^{-1}x \in \mathbb{Z}_N \mid \left(x \bmod \frac{N}{g} \right) \in J \right\}.$$

Expressing each $x \in \mathbb{Z}_N$ according to its division with remainder by N/g , i.e., $x = i \frac{N}{g} + x_0$ for $i = \lfloor (x/(N/g)) \rfloor \in [0..g-1]$ and $x_0 = x - i \frac{N}{g} \in [0.. \frac{N}{g} - 1]$, we get that

$$B = \left\{ \beta^{-1}(i \frac{N}{g} + x_0) \in \mathbb{Z}_N \mid x_0 \in J, i \in [0..g-1] \right\}.$$

■

Claim 9 For every index $\gamma \in \mathbb{Z}_N$, if $g \nmid \gamma$, then $\widehat{B}(\gamma) = 0$, and $\widehat{B}(\gamma) = \widehat{J_{N/g}}(\gamma\beta^{-1}/g)$ otherwise.

Proof: Fix $\gamma \in \mathbb{Z}_N$. By definition of the Fourier transform in \mathbb{Z}_N and the set B ,

$$\widehat{B}(\gamma) = \frac{1}{N} \sum_{x \in B} e(\gamma x/N) = \frac{1}{N} \sum_{x_0 \in J} \sum_{i \in [0..g-1]} e(\gamma \beta^{-1}(i \frac{N}{g} + x_0)/N)$$

where the last equality holds by Eq. 3. Rearranging this sum we get that

$$\begin{aligned} \widehat{B}(\gamma) &= \frac{1}{N} \sum_{x_0 \in J} e(\gamma \beta^{-1} x_0/N) \cdot \sum_{i \in [0..g-1]} e(\gamma \beta^{-1} i/g) \\ &= \begin{cases} 0 & g \nmid \gamma \beta^{-1} \\ \frac{g}{N} \sum_{x_0 \in J} e(\gamma \beta^{-1} x_0/N) & \text{otherwise} \end{cases} \end{aligned}$$

where the last equality holds since $\frac{1}{g} \sum_{i \in [0..g-1]} e(\gamma \beta^{-1} i/g) = 0$ if $g \nmid \gamma \beta^{-1}$, and it is 1 otherwise (see Sect. 3.1).

Note that $g \mid \gamma \beta^{-1}$ iff $g \mid \gamma$: This trivially holds for $g = 1$, and holds for $g > 1$, since $g \nmid \beta$ (because $g \mid N$ and $\gcd(N, \beta) = 1$). We conclude that $\widehat{B}(\gamma) = 0$ if $g \nmid \gamma$.

We focus next on the case that $g \mid \gamma$. Denote $\gamma' = \gamma/g$. Substituting γ with $\gamma'g$ in the above and rearranging, we get that, $\widehat{B}(\gamma) = \frac{g}{N} \sum_{x_0 \in J} e(\gamma' \beta^{-1} x_0/(N/g))$. This in turn is equal to $\widehat{J_{N/g}}(\gamma' \beta^{-1})$ (by definition of $\widehat{J_{N/g}}$). We conclude that $\widehat{B}(\gamma) = \widehat{J_{N/g}}(\gamma' \beta^{-1}) = \widehat{J_{N/g}}(\gamma \beta^{-1}/g)$ if $g \mid \gamma$. ■

Proposition 4 Fix any positive integers N, g s.t. $g \mid N$, and any $x \in \mathbb{Z}_N$ and $I = [s..t] \subseteq \mathbb{Z}_N$. Then $(gx \bmod N) \in I$ iff $(x \bmod \frac{N}{g}) \in \left[\left[\left(\frac{s}{g} \right) \right] .. \left[\left(\frac{t}{g} \right) \right] \right]$.

Proof: Observe that since $g \mid N$, then $(gx \bmod N) = (x \bmod \frac{N}{g}) \cdot g$. (Because $x = x_1 \frac{N}{g} + x_2$ for $x_1 < g, x_2 < \frac{N}{g}$, implying that $(x \bmod \frac{N}{g}) = x_2$ and $(gx \bmod N) = (x_1 N + gx_2 \bmod N) = gx_2$.) Thus, $(gx \bmod N) \geq s$ iff $(x \bmod \frac{N}{g}) \geq s/g$, which in turn happens iff $(x \bmod \frac{N}{g}) \geq \lceil (s/g) \rceil$ (because $x \bmod \frac{N}{g}$ is an integer). Similarly, $(gx \bmod N) \leq t$ iff $(x \bmod \frac{N}{g}) \leq \lfloor (t/g) \rfloor$. We conclude that $(gx \bmod N) \in I$ iff $(x \bmod \frac{N}{g}) \in \left[\left[\left(\frac{s}{g} \right) \right] .. \left[\left(\frac{t}{g} \right) \right] \right]$. ■

4.2 Approximating Distributions Uniform over Arithmetic Progressions

We show that there are explicit small sets approximating the uniform distribution over a given arithmetic progression.

Theorem 2 Part 2 (Approximating arithmetic progressions). For any integer $N > 0$, real $\varepsilon \in (0, 1)$, and a length $M \leq N$ arithmetic progression B in \mathbb{Z}_N , there is an explicit set $A \subseteq \mathbb{Z}_N$ of size polynomial in $1/\varepsilon$ and $(\ln M)^2$ that ε -approximates U_B in \mathbb{Z}_N .

Proof: Fix N, M and ε . We focus here on the case $B = [0..M-1]$ is an *interval* starting at zero; extensions to arbitrary arithmetic progressions appear in section 4.2.1.

Let $\varepsilon' = \varepsilon\rho^2/2$ for $\rho = \varepsilon/8\pi$. We show below that if $D_{A, \mathbb{Z}_M}(\mathcal{B}_{M,2}) < \varepsilon'$ (i.e., A has ε' -discrepancy on all rank 2 Bohr sets in \mathbb{Z}_M), then A ε -approximates U_B . This completes our proof, as by Theorem 2 Part 1 there exists sets A of size polynomial in $1/\varepsilon' = O(1/\varepsilon^3)$ and $(\ln M)^2$ s.t. $D_{A, \mathbb{Z}_M}(\mathcal{B}_{M,2}) < \varepsilon'$.

To relate approximating the distribution U_B over \mathbb{Z}_N to having ε' -discrepancy on all rank 2 Bohr sets in \mathbb{Z}_M , we do the following: For each $\alpha \in \mathbb{Z}_N$, denote by (q_α, r_α) its quotient and remainder in division-with-remainder by (the typically non-integer value) N/M . That is, q_α is the largest integer s.t. $q_\alpha \frac{N}{M} \leq \alpha$ and $r_\alpha = \alpha - q_\alpha \frac{N}{M}$ is the remainder. Write α as:

$$\alpha = N \left(\frac{q_\alpha}{M} + \frac{r_\alpha}{N} \right)$$

Rewrite each linear test $\chi_\alpha(x) = e(\alpha x/N)$ in \mathbb{Z}_N as:

$$\chi_\alpha(x) = e \left(\left(\left(\frac{q_\alpha x}{M} \right)_1 + \left(\frac{r_\alpha x}{N} \right)_1 \right) \right)$$

by replacing αx with $N \left(\frac{q_\alpha x}{M} + \frac{r_\alpha x}{N} \right)_1 = N \left(\left(\frac{q_\alpha x}{M} \right)_1 + \left(\frac{r_\alpha x}{N} \right)_1 \right)$. By Lemma 10, if $D_{A, \mathbb{Z}_M}(\mathcal{B}_{M,2}) < \varepsilon'$, then the joint distribution of pairs $\left(\left(\frac{q_\alpha x}{M} \right)_1, \left(\frac{r_\alpha x}{N} \right)_1 \right)$ with x drawn uniformly at random from A is “close” to their distribution with x drawn uniformly at random from B . Closeness is in the sense that for every two length ρ intervals $J_1, J_2 \subseteq [0, 1]$,

$$\left| \Pr_{x \in A} \left[\left(\frac{q_\alpha x}{M} \right)_1 \in J_1 \ \& \ \left(\frac{r_\alpha x}{N} \right)_1 \in J_2 \right] - \Pr_{x \in B} \left[\left(\frac{q_\alpha x}{M} \right)_1 \in J_1 \ \& \ \left(\frac{r_\alpha x}{N} \right)_1 \in J_2 \right] \right| < \varepsilon'$$

By Lemma 11, if the above equation holds, then A $(\frac{\varepsilon'}{\rho^2} + 4\pi\rho)$ -approximates U_B . Noting that $(\frac{\varepsilon'}{\rho^2} + 4\pi\rho) = \varepsilon$, we conclude that $D_{A, \mathbb{Z}_M}(\mathcal{B}_{M,2}) < \varepsilon'$ implies that A ε -approximates U_B . \blacksquare

Lemma 10 For any positive integers $M \leq N$, positive real ε and a subset $A \subseteq \mathbb{Z}_M$, if $D_{A, \mathbb{Z}_M}(\mathcal{B}_{M,2}) < \varepsilon'$, then for all integers $\alpha \in [0, M)$, reals $\beta \in [0, N/M)$, and intervals $J_1, J_2 \subseteq [0, 1]$,

$$\left| \Pr_{x \in A} \left[\left(\frac{\alpha x}{M} \right)_1 \in J_1 \ \text{and} \ \left(\frac{\beta x}{N} \right)_1 \in J_2 \right] - \Pr_{x \in B} \left[\left(\frac{\alpha x}{M} \right)_1 \in J_1 \ \text{and} \ \left(\frac{\beta x}{N} \right)_1 \in J_2 \right] \right| < \varepsilon' \quad (4)$$

Remark. In particular, Eq. 4 holds for any $\alpha, \beta, J_1, J_2, \varepsilon'$ s.t. $(\alpha, \beta) = (q_z, r_z)$ are the quotient and remainder in the division-with-remainder by N/M of some $z \in \mathbb{Z}_N$, $J_1, J_2 \subseteq [0, 1]$ and length ρ intervals (because q_z is an integer in $[0, M)$ and r_z is a real in $[0, N/M)$).

Proof: Fix parameters α, β, J_1, J_2 and ε' for Eq. 4. Fix $A \subseteq \mathbb{Z}_M$ s.t. $D_{A, \mathbb{Z}_M}(\mathcal{B}_{M,2}) < \varepsilon'$. In the following we first show that Eq. 4 above holds iff Eq. 5 below holds. We then argue that Eq. 5 holds as it bounds the discrepancy of A on a rank 2 Bohr set in \mathbb{Z}_M . We conclude that Eq. 4 holds.

We map the sets $\left(\frac{\alpha B}{M} \right)_1$ and $\left(\frac{\beta B}{N} \right)_1$ into \mathbb{Z}_M by the one-to-one mappings:

$$\begin{aligned} (i) \quad & \left(\frac{\alpha x}{M} \right)_1 \mapsto \alpha x \bmod M \\ (ii) \quad & \left(\frac{\beta x}{N} \right)_1 \mapsto x \end{aligned}$$

(The latter map is a one-to-one because $\beta < N/M$ and hence for $x \in B = [0..M-1]$, $\left(\frac{\beta x}{N} \right)_1$ does not wrap around 0.) Denote by \bar{J}_1 and \bar{J}_2 the intervals in \mathbb{Z}_M that are the images of the sets $J'_1 = J_1 \cap \left(\frac{\alpha B}{M} \right)_1$ and $J'_2 = J_2 \cap \left(\frac{\beta B}{N} \right)_1$ under mappings (i) and (ii), respectively. Since mappings (i),(ii) are one-to-one, then Eq. 4 holds iff the following holds:

$$\left| \Pr_{x \in A} [(\alpha x \bmod M) \in \bar{J}_1 \ \text{and} \ x \in \bar{J}_2] - \Pr_{x \in B} [(\alpha x \bmod M) \in \bar{J}_1 \ \text{and} \ x \in \bar{J}_2] \right| < \varepsilon' \quad (5)$$

Observing that the left hand side of Eq. 5 is the discrepancy of A on the rank 2 Bohr set

$$R = \{x \in \mathbb{Z}_M \mid (\alpha x \bmod M) \in \bar{J}_1 \text{ and } x \in \bar{J}_2\},$$

we conclude that Eq. 5 holds and hence Eq. 4 holds. \blacksquare

Lemma 11 *If Eq. 4 holds for every integer $\alpha \in [0, M)$, real $\beta \in [0, N/M)$, and length ρ intervals $J_1, J_2 \subseteq [0, 1]$, then A ($\frac{\varepsilon}{\rho^2} + 4\pi\rho$)-approximates U_B in \mathbb{Z}_N .*

Proof Sketch. Intuitively, if the distribution of pairs $((\frac{q_\alpha x}{M})_1, (\frac{r_\alpha x}{N})_1)$ with x drawn from U_A is “close” to the distribution of such pairs with x drawn from U_B , then the distribution of sums $\chi_\alpha(x) = ((\frac{q_\alpha x}{M})_1 + (\frac{r_\alpha x}{N})_1)_1$ over $x \in A$ is “close” to the distribution of such sums over $x \in B$. Namely, the expected value of $\chi_\alpha(x)$ with x drawn from U_A is “close” to its expected value with x drawn from U_B .

Capturing this intuition requires some technical work, where we rewrite $\mathbb{E}_{x \in S} \chi_\alpha(x)$, $S = A, B$, as a sum over expressions depending on pairs of intervals J_1, J_2 , and use Eq. 4 (and other manipulations) to bound these expressions. Details are omitted from this extended abstract. \blacksquare

4.2.1 From Approximating Intervals to Approximating Arithmetic Progressions

We show that given any length M arithmetic progression $P = P_{\alpha, [s..s+M-1]}$ and any set A that ε -approximates $U_{[0..M-1]}$ in \mathbb{Z}_N , the set $A' = \alpha(A + s)$ is a set of size $|A'| \leq |A|$ that ε -approximates U_P .

Lemma 12 *For any positive integers $M \leq N$ and any length M arithmetic progression $P = P_{\alpha, [s..M-1+s]}$ in \mathbb{Z}_N , if $A \subseteq \mathbb{Z}_N$ ε -approximates $U_{[0..M-1]}$, then $A' \stackrel{\text{def}}{=} \alpha(A + s)$ ε -approximates U_P .*

Proof: For any $\beta \in \mathbb{Z}_N$ and subsets $S, S' \subseteq \mathbb{Z}_N$, denote $\text{diff}_\beta(S, S') \stackrel{\text{def}}{=} \mathbb{E}_{x \in S} e(\beta x/N) - \mathbb{E}_{x \in S'} e(\beta x/N)$. We prove that $|\text{diff}_\beta(A', P)| < \varepsilon$ for all $\beta \in \mathbb{Z}_N$. Fix β . By definition of A' and P , $|\text{diff}_\beta(A', P)| = |\mathbb{E}_{x \in A} e(\beta\alpha(x+s)/N) - \mathbb{E}_{x \in [0..M-1]} e(\beta\alpha(x+s)/N)|$. The latter is equal to $|e(\beta\alpha s/N)| |\mathbb{E}_{x \in A} e(\beta\alpha x/N) - \mathbb{E}_{x \in [0..M-1]} e(\beta\alpha x/N)| = 1 \cdot |\text{diff}_{\alpha\beta}(A, [0..M-1])| < \varepsilon$ where the last inequality holds, because A ε -approximates $U_{[0..M-1]}$ means that $|\text{diff}_{\beta'}(A, [0..M-1])| < \varepsilon$ for all $\beta' \in \mathbb{Z}_N$. \blacksquare

5 Deterministically Finding Significant Fourier Coefficients

We present our deterministic and robust SFT algorithm, and prove Theorem 1.

5.1 The SFT Algorithm

At a high level, our SFT algorithm is a binary search algorithm that repeatedly: (1) Partitions the set of potentially significant Fourier coefficients into two subsets. (2) Tests each subset to decide if it (potentially) contains a significant Fourier coefficient. (3) Continues recursively on any subset with a positive test result.

Elaborating on the above, at each step of this search, the set of potentially significant Fourier coefficients is maintained as a collection \mathcal{J} of intervals, starting with \mathcal{J} containing the four intervals $[i \frac{N}{4}..(i+1) \frac{N}{4}]$, $i = 0, \dots, 3$. To maintain efficiency, the intervals $[a..b]$ are represented by their endpoints pairs $\{a, b\}$. The partition step partitions every interval $J = [a..b] \in \mathcal{J}$ into its lower and upper halves: $J_1 = [a..c]$ and $J_2 = [c+1..b]$ for $c = \lfloor ((a+b)/2) \rfloor$ its center. For $i = 1, 2$, the test estimates the Fourier weight of f on J_i , denoted $\hat{f}(J_i)^2 \stackrel{\text{def}}{=} \sum_{\alpha \in J_i} |\hat{f}(\alpha)|^2$, returning YES if this weight exceeds the significance threshold τ .

The set \mathcal{J} is updated by removing J , and inserting each J_i ($i = 1, 2$) iff it passes the test. After $O(\log N)$ steps this search terminates with a collection \mathcal{J} of length one intervals containing the frequencies of the (potentially) significant Fourier coefficients. The algorithm wraps up by executing a sieving step, where for each frequency α of a potentially significant Fourier coefficient, a $O(\tau)$ -approximation for $\hat{f}(\alpha)$ is computed: $val_\alpha = \frac{1}{|A|} \sum_{x \in A-y} f(x) \overline{\chi_\alpha(x)}$ (for an arbitrary $y \in \cup_{\ell=1}^{\lfloor \log N \rfloor} B_\ell$); and the algorithm outputs the pairs (α, val_α) for all α found to be significant, i.e., $val_\alpha \geq \tau/2$.

The heart of the algorithm is the test deciding which intervals potentially contain a significant Fourier coefficient. This test, given an interval $J = [a..b]$, answers YES if the Fourier weight on J , exceeds the significance threshold τ , and answers NO if the Fourier weight of a slightly larger interval $J' \supseteq J$ is less than $\tau/2$. This is achieved by estimating the ℓ_2 norm (i.e., sum of squared Fourier coefficients) of a filtered version of the input function f , when using a filter h that passes Fourier coefficients in J and decays fast outside of J . The filters we use are functions $h = h_{\ell, c}$ which are (normalized) periodic square functions with

support size 2^ℓ when phase shifted by $-c$, for $c = \lfloor ((a+b)/2) \rfloor$ the center of J and $\ell = \lfloor (\log \frac{N}{4(b-a)}) \rfloor$ a function of J 's length:

$$h_{\ell,c}(y) \stackrel{\text{def}}{=} \begin{cases} \frac{N}{2^\ell} \cdot \chi_{-c}(y) & y \in [0..2^\ell - 1] \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The filter $h_{\ell,c}$ passes all frequencies that lie within the length $N/2^{\ell+2}$ interval J centered around c , and decays fast outside of J . The filtered version of f is $f * h$, and we estimate its ℓ_2 norm $\|f * h\|_2^2$ by the estimator:

$$\text{est}_{\ell,c}(f) \stackrel{\text{def}}{=} \frac{1}{|A|} \sum_{x \in A} \left(\frac{1}{|B_\ell|} \sum_{y \in B_\ell} \chi_{-c}(y) \overline{f(x-y)} \right)^2 \quad (7)$$

for A a small biased set in \mathbb{Z}_N , and B_ℓ approximating the uniform distribution over $[0..2^\ell - 1]$ in \mathbb{Z}_N .

A pseudo-code of the algorithm follows.

Algorithm 5 SFT.

Input: $N \in \mathbb{N}$, $\tau \in (0, 1]$, oracle access to $f: \mathbb{Z}_N \rightarrow \mathbb{C}$.

1. *Initialize:* $L = \emptyset$; $\mathcal{J} \leftarrow \left\{ \left\{ i \frac{\lambda N}{2}, (i+1) \frac{\lambda N}{2} \right\}_{i=0}^{\frac{2}{\lambda}-1} \text{ for } \lambda = 1/2; \gamma = \tau / (49t^2 \ln N) \right.$
2. *Construct queries:*
 - $A := \text{AIKPS}(N, \gamma)$
 - For $\ell = 1, \dots, \log N$, $B_\ell := \text{AIKPS}(2^\ell, \gamma_\ell)$ for $\gamma_\ell := \gamma^3 / (128\pi^2 \ell^2)$

Remark: $\text{AIKPS}(N, \varepsilon)$ is as specified in Sect. 3.2, Eq. 1.
3. *Main Loop:* While $\exists \{a, b\} \in \mathcal{J}$ s.t. $b \neq a$ do:
 - (a) Remove $\{a, b\}$ from \mathcal{J} , denote $c' = \lfloor ((a+b)/2) \rfloor$
 - (b) For each pair $\{a', b'\}$ out of the pairs $\{a, c'\}$ and $\{c'+1, b\}$ do:
 - i. Let $\ell = \lfloor (\log \frac{\lambda N}{2(b'-a')}) \rfloor$ and $c = \lfloor ((a'+b')/2) \rfloor$
 - ii. Compute $\text{est}_{\ell,c} \leftarrow \frac{1}{|A|} \sum_{x \in A} \left(\frac{1}{|B_\ell|} \sum_{y \in B_\ell} \chi_{-c}(y) \overline{f(x-y)} \right)^2$
 - iii. If $\text{est}_{\ell,c} \geq \tau/2$, insert $\{a', b'\}$ to \mathcal{J}
4. *Sieving:* For each $\{a, b\} \in \mathcal{J}$, and each $\alpha \in [a..b]$
 - (a) Compute $\text{val}(\alpha) \leftarrow \left| \frac{1}{|A|} \sum_{x \in A} \chi_\alpha(x) \overline{f(x)} \right|^2$
 - (b) If $\text{val}(\alpha) \geq \tau/2$, insert α to L
5. Output $\{(\alpha, \text{val}_\alpha)\}_{\alpha \in L}$

Remarks. (1) To keep this paper self contained, in Step 2 of the pseudo-code we use the small biased sets of [AIK⁺90] which were specified in Sect. 3.2, Eq. 1. Nevertheless, any other construction of small biased sets may be used (with set sizes $|A|, |B_\ell|$ varying accordingly). (2) To simplify notations, we assume without loss of generality that $0 \in \cup_\ell B_\ell$ (otherwise add it), and $\|f\|_2 = 1$ (otherwise normalize f by dividing each read value by an energy estimator $\frac{1}{|A|} \sum_{x \in A} \overline{f(x)}^2$).

5.2 Proof of Theorem 1

The proof of Theorem 1 is simple given our new result on explicit sets approximating given arithmetic progressions in \mathbb{Z}_N (Theorems 2 and 4) together with the work of [Aka09]:

Proof:[Proof of Theorem 1.] Our algorithm builds on the algorithm of [Aka09] while replacing their randomized construction of sets $S = \bigcup_\ell (A - B_\ell)$ with a deterministic construction. Our deterministic construction produces a set S which is (N, τ, t) -good (see Corollary 13 below). When S is (N, τ, t) -good, the analysis of [Aka09] shows that the algorithm succeeds (see Theorem 14 below). Namely, the algorithm outputs $L \supseteq \text{Heavy}_\tau(f)$ (with probability at least $1 - 1/N^{\Theta(1)}$ over the random noise η) in running time is $O(\frac{1}{\tau^2} \log N \cdot |S|)$. Finally, this running time polynomial in $\log N, 1/\tau$ and t by the definition of good sets. ■

As a corollary of our results on explicit constructions (Theorems 2,4), the queries constructed in our SFT algorithm are (N, τ, t) -good.

Corollary 13 The set $S = \bigcup_{\ell}(A - B_{\ell})$ for A, B_{ℓ} as in Algorithm 5 is an (N, τ, t) -good set.

Proof: The set S is good, because: (i) A is a γ -biased set in \mathbb{Z}_N . (ii) B_{ℓ} γ -approximates $U_{[0..2^{\ell}-1]}$ (by Theorem 4 and the fact that B_{ℓ} is a $\gamma_{\ell} = \gamma^3/(128\pi^2\ell^2)$ -biased set in $\mathbb{Z}_{2^{\ell}}$). Finally, $|A|, |B_{\ell}|$ are polynomial in $\log N$ and $1/\gamma = O(t^2 \ln N/\tau)$. ■

Theorem 14 ([Aka09]) If the queries $S = \bigcup_{\ell}(A - B_{\ell})$ for A, B_{ℓ} as in Algorithm 5 are (N, τ, t) -good, then the following holds. Given N, τ, t , and $\{(x, f^i(x))\}_{x \in S}$ for $f' = f + \eta$ a complex valued function over \mathbb{Z}_N s.t. $L_1(\hat{f}) \leq t$ and η is a $\tau/3$ -random noise, Algorithm 5 outputs $\{(\alpha, \text{val}_{\alpha})\}_{\alpha \in L}$ s.t. (with probability at least $1 - 1/N^{\Theta(1)}$ over the random noise η) $L \supseteq \text{Heavy}_{\tau}(f)$ and val_{α} are $O(\tau)$ -approximations of $\hat{f}(\alpha)$. The running time of the algorithm is $O(\frac{1}{\tau^2} \log N \cdot |S|)$.

6 Conclusions

We presented the first deterministic SFT algorithm for functions over \mathbb{Z}_N , which is: (1) Local, i.e., its running time is polynomial in $\log N, 1/\tau$ and $L_1(\hat{f})$; and (2) Robust to random noise. Our main technical novelty lies in proving that there exists explicit constructions of small sets with small discrepancy in all rank d Bohr sets, as well as small sets approximating the uniform distribution over a given arithmetic progression.

Extensions. Our deterministic SFT algorithm extends to handle functions over all finite abelian groups G (given by their generators g_1, \dots, g_k and their orders N_1, \dots, N_k). This extension is motivated for example by functions over domains $G = \mathbb{Z}_{N_1} \times \dots \times \mathbb{Z}_{N_k}$ arising in image/video processing applications ($k = 2, 3$) and machine learning applications (large k). As a central ingredient for this extension we present explicit small sets approximating the uniform distribution over rectangles $R_{t,\ell} = [0..N_1] \times \dots \times [0..N_{t-1}] \times [0..2^{\ell} - 1] \times \{0\}^{k-t}$ for given $t \in [k]$ and $\ell \in [\log N_t]$. Details are omitted from this extended abstract.

Applications to sparse Fourier approximation, compressed sensing and sketching. Using our SFT algorithm we obtain deterministic and robust algorithms for sparse Fourier approximation, compressed sensing and sketching. Our algorithms, given N, m, ε, t and oracle access to a signal $x \in \mathbb{C}^N$ s.t. $L_1(\hat{x}) \leq t$ (resp., a sketch Ax for $A = A_{N,m,\varepsilon,t} \in \mathbb{C}^{\text{poly}(\log N, m/\varepsilon, t)} \times N$ an explicit measurement matrix), output a near optimal m -sparse approximation R of x , i.e., $\|x - R\|_2^2 \leq \|x - R_{\text{opt}}\|_2^2 + \varepsilon$ for R_{opt} the best m -terms approximation of x in the Fourier (resp., standard) basis. Our algorithms are robust to $m/3\varepsilon$ -random noise, and their running time is polynomial in $\log N, m/\varepsilon$ and t . Given our SFT algorithm, the derivation of these algorithms is standard; details are omitted from this extended abstract.

Open questions. Our results on explicit constructions yields sets sizes that are efficient but not optimal: Probabilistic method arguments show that there are randomized constructions of sets of size $O(d \log N/\varepsilon^2)$ with ε -discrepancy in all rank d Bohr sets, as well as sets of size $O((\log N)/\varepsilon^2)$ ε -approximating the uniform distribution over a given arithmetic progression. Whether these bounds can be matched by explicit constructions is an open problem.

Acknowledgments

The author is most grateful to Noga Alon, Shafi Goldwasser, Venkat Guruswami, Piotr Indyk, Mark Iwen, Martin Strauss and Avi Wigderson for helpful comments and discussions.

References

- [AGS03] A. Akavia, S. Goldwasser, and S. Safra. Proving Hard-Core Predicates using List Decoding. In *Proc. of 44th IEEE Annual Symposium on Foundations of Computer Science (FOCS'03)*, pages 146–157. IEEE Computer Society, 2003.
- [AIK⁺90] M. Ajtai, H. Iwaniec, J. Komlos, J. Pintz, and E. Szemerédi. Constructions of a this set with small fourier coefficients. *Bull. London Math. Soc.*, 22:583–590, 1990.
- [Aka09] A. Akavia. Solving Hidden Number Problem with One Bit Oracle and Advice. In *proceedings of the 29th Annual International Cryptology Conference (Crypto'09)*, 2009.
- [AM95] N. Alon and Y. Mansour. ε -discrepancy sets and their application for interpolation of sparse polynomials. *IPL: Information Processing Letters*, 54, 1995.
- [CM05] G. Cormode and S. Muthukrishnan. Towards an algorithmic theory of compressed sensing. In *DIMACS TR: 2005-25*, 2005.
- [CM06] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In Paola Flocchini and Leszek Gasieniec, editors, *SIROCCO*, volume 4056 of *Lecture Notes in Computer Science*, pages 280–294. Springer, 2006.

- [CRT06] E. J. Candes, J. K. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.
- [CT65] J.W. Cooley and J.W. Tukey. An algorithm for machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, Apr 1965.
- [DeV07] R. A. DeVore. Deterministic constructions of compressed sensing matrices. *J. Complex.*, 23(4-6):918–925, 2007.
- [Don05] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 42(4):1289–1306, April, 2005.
- [GGI⁺02] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse fourier representations via sampling. In *Proc. of 34 ACM Annual Symposium on Theory of Computing (STOC'02)*, pages 152–161. ACM Press, 2002.
- [GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc. 27th ACM Annual Symposium on Theory of Computing (STOC'89)*, pages 25–32, 1989.
- [GLR08] V. Guruswami, J. R. Lee, and A. Razborov. Almost euclidean subspaces of ℓ_1 via expander codes. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 353–362, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.
- [GM06] Sumit Ganguly and Anirban Majumder. Cr-precise: A deterministic summary structure for update data streams. *CoRR*, abs/cs/0609032, 2006.
- [GMS05] A. C. Gilbert, S. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal sparse fourier representation via sampling. In *in Proc. SPIE Wavelets XI*, 2005.
- [GSTV06] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. Algorithmic linear dimension reduction in the l_1 norm for sparse vectors. *CoRR*, abs/cs/0608079, 2006.
- [Ind07] P. Indyk. Sketching, streaming and sub-linear space algorithms. graduate course notes, available at <http://stellar.mit.edu/s/course/6/fa07/6,895/>, 2007.
- [Ind08] P. Indyk. Explicit constructions for compressed sensing of sparse signals. In *SODA'08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 30–33, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.
- [IS08] M. A. Iwen and C. V. Spencer. Improved bounds for a deterministic sublinear-time sparse fourier algorithm. In *Conference on Information Sciences and Systems (CISS), Princeton, NJ*, 2008.
- [Iwe07] M. A. Iwen. A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. *CoRR*, abs/0708.1211, 2007.
- [Iwe08] M. A. Iwen. A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 20–29, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.
- [Kat89] M. Katz. An estimate for characters sum. *J. AMS*, 2(2):197–200, 1989.
- [KM93] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SICOMP*, 22(6):1331–1348, 1993.
- [Man95] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM J. on Computing*, 24(2):357–368, 1995.
- [Mut03] S. Muthukrishnan. Data streams: Algorithm and applications (invited talk at soda'03). available at <http://athos.rutgers.edu/muthu/stream-1-1.ps>, 2003.
- [RSW93] A. Razborov, E. Szemerédi, and A. Wigderson. Constructing small sets that are uniform in arithmetic progressions. *Combinatorics, Probability & Computing*, 2:513–518, 1993.
- [TV06] Terence Tao and Van H. Vu. *Additive Combinatorics*. Series: Cambridge Studies in Advanced Mathematics (No. 105), September 2006.

Forest Density Estimation

Anupam Gupta[†], John Lafferty^{†*}, Han Liu^{†*}, Larry Wasserman^{*†}, Min Xu[†]

[†]School of Computer Science

^{*}Department of Statistics

Carnegie Mellon University

Abstract

We study graph estimation and density estimation in high dimensions, using a family of density estimators based on forest structured undirected graphical models. For density estimation, we do not assume the true distribution corresponds to a forest; rather, we form kernel density estimates of the bivariate and univariate marginals, and apply Kruskal's algorithm to estimate the optimal forest on held out data. We prove an oracle inequality on the excess risk of the resulting estimator relative to the risk of the best forest. For graph estimation, we consider the problem of estimating forests with restricted tree sizes. We prove that finding a maximum weight spanning forest with restricted tree size is NP-hard, and develop an approximation algorithm for this problem. Viewing the tree size as a complexity parameter, we then select a forest using data splitting, and prove bounds on excess risk and structure selection consistency of the procedure. Experiments with simulated data and microarray data indicate that the methods are a practical alternative to sparse Gaussian graphical models.

1 Introduction

One way to explore the structure of a high dimensional distribution P for a random vector $X = (X_1, \dots, X_d)$ is to estimate its undirected graph. The undirected graph G associated with P has d vertices corresponding to the variables X_1, \dots, X_d , and omits an edge between two nodes X_i and X_j if and only if X_i and X_j are conditionally independent given the other variables. Currently, the most popular methods for estimating G assume that the distribution P is Gaussian. Finding the graphical structure in this case amounts to estimating the inverse covariance matrix Ω ; the edge between X_j and X_k is missing if and only if $\Omega_{jk} = 0$. Algorithms for optimizing the ℓ_1 -regularized log-likelihood have recently been proposed that efficiently produce sparse estimates of the inverse covariance matrix and the underlying graph (Banerjee et al., 2008; Friedman et al., 2007).

In this paper our goal is to relax the Gaussian assumption and to develop nonparametric methods for estimating the graph of a distribution. Of course, estimating a high dimensional distribution is impossible without making any assumptions. The approach we take here is to force the graphical structure to be a forest, where each pair of vertices is connected by at most one path. Thus, we relax the distributional assumption of normality but we restrict the family of undirected graphs that are allowed.

If the graph for P is a forest, then its density p can be written as

$$p(x) = \prod_{(i,j) \in E} \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \prod_{k=1}^d p(x_k) \quad (1.1)$$

where E is the set of edges in the forest. Thus, it is only necessary to estimate the bivariate and univariate marginals. Given any distribution P with density p , there is a tree T and a density p_T whose graph is T and which is closest in Kullback-Leibler divergence to p . When P is known, then the best fitting tree distribution can be obtained by Kruskal's algorithm (Kruskal, 1956), or other algorithms for finding a maximum weight spanning tree. In the discrete case, the algorithm can be applied to the estimated probability mass function, resulting in a procedure originally proposed by Chow and Liu (1968). Here we are concerned with continuous random variables, and we estimate the bivariate marginals with nonparametric kernel density estimators.

In high dimensions, fitting a fully connected spanning tree can be expected to over fit. We regulate the complexity of the forest by selecting the edges to include using a data splitting scheme, a simple form of

cross validation. In particular, we consider the family of forest structured densities that use the marginal kernel density estimates constructed on the first partition of the data, and estimate the risk of the resulting densities over a second, held out partition. The optimal forest in terms of the held out risk is then obtained by finding a maximum weight spanning forest for an appropriate set of edge weights.

While tree and forest structured density estimation has been long recognized as a useful tool, there has been little theoretical analysis of the statistical properties of such density estimators. The main contribution of this paper is an analysis of the asymptotic properties of forest density estimation in high dimensions. We allow both the sample size n and dimension d to increase, and prove oracle results on the risk of the method. In particular, we assume that the univariate and bivariate marginal densities lie in a Hölder class with exponent β (see Section 4 for details), and show that

$$R(\widehat{p}_{\widehat{F}}) - \min_F R(\widehat{p}_F) = O_P \left(\sqrt{\log(nd)} \left(\frac{k^* + \widehat{k}}{n^{\beta/(2+2\beta)}} + \frac{d}{n^{\beta/(1+2\beta)}} \right) \right) \quad (1.2)$$

where R denotes the risk, the expected negative log-likelihood, \widehat{k} is the number of edges in the estimated forest \widehat{F} , and k^* is the number of edges in the optimal forest F^* that can be constructed in terms of the kernel density estimates \widehat{p} .

Among the only other previous work analyzing tree structured graphical models is Tan et al. (2009a) and Chechetka and Guestrin (2007). Tan et al. (2009a) analyze the error exponent in the rate of decay of the error probability for estimating the tree, in the fixed dimension setting, and Chechetka and Guestrin (2007) give a PAC analysis. An extension to the Gaussian case is given by Tan et al. (2009b).

In addition to the above results on risk consistency, we also study the problem of estimating forests with restricted tree sizes. In many applications, one is interested in obtaining a graphical representation of a high dimensional distribution to aid in interpretation. For instance, a biologist studying gene interaction networks might be interested in a visualization that groups together genes in small sets. Such a clustering approach through density estimation is problematic if the graph is allowed to have cycles, as this can require marginal densities to be estimated with many interacting variables. Restricting the graph to be a forest beats the curse of dimensionality by requiring only univariate and bivariate marginal densities. To group the variables into small interacting sets, we are led to the problem of estimating a maximum weight spanning forest with a restriction on the size of each component tree. As we demonstrate, estimating restricted tree size forests can also be useful in model selection for the purpose of risk minimization. Limiting the tree size gives another way of regulating tree complexity that provides larger family of forest to select from in the data splitting procedure.

While the problem of finding a maximum weight forest with restricted tree size may be natural, it appears not to have been studied previously. We prove that the problem is NP-hard through a reduction from the problem of Exact 3-Cover (Garey & Johnson, 1979), where we are given a set X and a family \mathcal{S} of 3-element subsets of X , and must choose a subfamily of disjoint 3-element subsets to cover X . While finding the exact optimum is hard, we give a practical 4-approximation algorithm for finding the optimal tree restricted forest; that is, our algorithm outputs a forest whose weight is guaranteed to be at least $\frac{1}{4}w(F^*)$, where $w(F^*)$ is the weight of the optimal forest. This approximation guarantee translates into excess risk bounds on the constructed forest using our previous analysis, as the weight of the forest corresponds to contribution to the risk coming from the bivariate marginals over the edges in the forest. Our experimental results with this approximation algorithm show that it can be effective in practice for forest density estimation.

In Section 2 we review some background and notation. In Section 3 we present a two-stage algorithm, and we provide a theoretical analysis of the algorithm in Section 4, with the detailed proofs collected in the full arXiv version of this paper (Liu et al., 2010). In Section 6 we present experiments with both simulated data and gene microarray data, where the problem is to estimate the gene-gene association graph, which has been previously studied using Gaussian graphical models by Wille et al. (2004).

2 Preliminaries and Notation

Let $p^*(x)$ be a probability density with respect to Lebesgue measure $\mu(\cdot)$ on \mathbb{R}^d and let $X^{(1)}, \dots, X^{(n)}$ be n independent identically distributed \mathbb{R}^d -valued data vectors sampled from $p^*(x)$ where $X^{(i)} = (X_1^{(i)}, \dots, X_d^{(i)})$. Let \mathcal{X}_j denote the range of $X_j^{(i)}$ and let $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_d$.

A graph is a forest if it is acyclic. If F is a d -node undirected forest with vertex set $V_F = \{1, \dots, d\}$ and edge set $E(F) \subset \{1, \dots, d\} \times \{1, \dots, d\}$, the number of edges satisfies $|E(F)| < d$, noting that we do not restrict the graph to be connected. We say that a probability density function $p(x)$ is *supported by a forest* F if the density can be written as

$$p_F(x) = \prod_{(i,j) \in E(F)} \frac{p(x_i, x_j)}{p(x_i) p(x_j)} \prod_{k \in V_F} p(x_k), \quad (2.1)$$

where each $p(x_i, x_j)$ is a bivariate density on $\mathcal{X}_i \times \mathcal{X}_j$, and each $p(x_k)$ is a univariate density on \mathcal{X}_k (Lauritzen, 1996).

Let \mathcal{F}_d be the family of forests with d nodes, and let \mathcal{P}_d be the corresponding family of densities:

$$\mathcal{P}_d = \left\{ p \geq 0 : \int_{\mathcal{X}} p(x) d\mu(x) = 1, \text{ and } p(x) \text{ satisfies (2.1) for some } F \in \mathcal{F}_d \right\}. \quad (2.2)$$

To bound the number of labeled spanning forests on d nodes, note that each such forest can be obtained by forming a labeled tree on $d + 1$ nodes, and then removing node $d + 1$. From Cayley's formula (Cayley, 1889; Aigner & Ziegler, 1998), we then obtain the following.

Proposition 2.1 *The size of the collection \mathcal{F}_d of labeled forests on d nodes satisfies*

$$|\mathcal{F}_d| < (d + 1)^{d-1}. \quad (2.3)$$

Define the oracle forest density

$$q^* = \arg \min_{q \in \mathcal{P}_d} D(p^* \| q) \quad (2.4)$$

where the Kullback-Leibler divergence $D(p \| q)$ between two densities p and q is

$$D(p \| q) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx, \quad (2.5)$$

under the convention that $0 \log(0/q) = 0$, and $p \log(p/0) = \infty$ for $p \neq 0$. The following is proved by Bach and Jordan (2003).

Proposition 2.2 *Let q^* be defined as in (2.4). There exists a tree $T^* \in \mathcal{F}_d$, such that*

$$q^* = p_{T^*}^* = \prod_{(i,j) \in E(T^*)} \frac{p^*(x_i, x_j)}{p^*(x_i) p^*(x_j)} \prod_{k \in V_{T^*}} p^*(x_k) \quad (2.6)$$

where $p^*(x_i, x_j)$ and $p^*(x_i)$ are the bivariate and univariate marginal densities of p^* .

For any density $q(x)$, the negative log-likelihood risk $R(q)$ is defined as

$$R(q) = -\mathbb{E} \log q(X) = - \int_{\mathcal{X}} p^*(x) \log q(x) dx. \quad (2.7)$$

It is straightforward to see that the density q^* defined in (2.4) also minimizes the negative log-likelihood loss:

$$q^* = \arg \min_{q \in \mathcal{P}_d} D(p^* \| q) = \arg \min_{q \in \mathcal{P}_d} R(q) \quad (2.8)$$

We thus define the oracle risk as $R^* = R(q^*)$. Using Proposition 2.2 and equation (2.1), we have

$$\begin{aligned} R^* &= R(q^*) = R(p_{T^*}^*) \\ &= - \int_{\mathcal{X}} p^*(x) \left(\sum_{(i,j) \in E(T^*)} \log \frac{p^*(x_i, x_j)}{p^*(x_i) p^*(x_j)} + \sum_{k \in V_{T^*}} \log(p^*(x_k)) \right) dx \\ &= - \sum_{(i,j) \in E(T^*)} \int_{\mathcal{X}_i \times \mathcal{X}_j} p^*(x_i, x_j) \log \frac{p^*(x_i, x_j)}{p^*(x_i) p^*(x_j)} dx_i dx_j - \sum_{k \in V_{T^*}} \int_{\mathcal{X}_k} p^*(x_k) \log p^*(x_k) dx_k \\ &= - \sum_{(i,j) \in E(T^*)} I(X_i; X_j) + \sum_{k \in V_{T^*}} H(X_k), \end{aligned} \quad (2.9)$$

where

$$I(X_i; X_j) = \int_{\mathcal{X}_i \times \mathcal{X}_j} p^*(x_i, x_j) \log \frac{p^*(x_i, x_j)}{p^*(x_i) p^*(x_j)} dx_i dx_j \quad (2.10)$$

is the mutual information between the pair of variables X_i, X_j and

$$H(X_k) = - \int_{\mathcal{X}_k} p^*(x_k) \log p^*(x_k) dx_k \quad (2.11)$$

is the entropy. While the best forest will in fact be a spanning tree, the densities $p^*(x_i, x_j)$ are in practice not known. We estimate the marginals using finite data, in terms of a kernel density estimates $\hat{p}_{n_1}(x_i, x_j)$ over a training set of size n_1 . With these estimated marginals, we consider all forest density estimates of the form

$$\hat{p}_F(x) = \prod_{(i,j) \in E(F)} \frac{\hat{p}_{n_1}(x_i, x_j)}{\hat{p}_{n_1}(x_i) \hat{p}_{n_1}(x_j)} \prod_{k \in V_F} \hat{p}_{n_1}(x_k). \quad (2.12)$$

Within this family, the best density estimate may not be supported on a full spanning tree, since a full tree will in general be subject to over fitting. Analogously, in high dimensional linear regression, the optimal regression model will generally be a full p -dimensional fit, with a nonzero parameter for each variable. However, when estimated on finite data the variance of a full model will dominate the squared bias, resulting in over fitting. In our setting of density estimation we will regulate the complexity of the forest by cross validating over a held out set.

There are several different ways to judge the quality of a forest structured density estimator. In this paper we concern ourselves with prediction and density estimation, and thus focus on risk consistency.

Definition 2.3 ((Risk consistency)) For an estimator $\hat{q}_n \in \mathcal{P}_d$, the excess risk is defined as $R(\hat{q}_n) - R^*$. The estimator \hat{q}_n is risk consistent with convergence rate δ_n if

$$\lim_{M \rightarrow \infty} \limsup_{n \rightarrow \infty} \mathbb{P}(R(\hat{q}_n) - R^* \geq M\delta_n) = 0. \quad (2.13)$$

In this case we write $R(\hat{q}_n) - R^* = O_P(\delta_n)$.

It is important to note that this criterion is an oracle property, in the sense that the true density $p^*(x)$ is not restricted to be supported by a tree; rather, the property assesses how well a given estimator \hat{q} approximates the best forest density (the oracle) within a class.

3 Kernel Density Estimation For Forests

If the true density $p^*(x)$ were known, by Proposition 2.2, the density estimation problem would be reduced to finding the best tree structure T_d^* , satisfying

$$T_d^* = \arg \min_{T \in \mathcal{T}_d} R(p_T^*) = \arg \min_{T \in \mathcal{T}_d} D(p^* \| p_T^*). \quad (3.1)$$

The optimal tree T_d^* can be found by minimizing the right hand side of (2.9). Since the entropy term $H(X) = \sum_k H(X_k)$ is constant across all trees, this can be recast as the problem of finding the maximum weight spanning tree for a weighted graph, where the weight of the edge connecting nodes i and j is $I(X_i; X_j)$. Kruskal's algorithm (Kruskal, 1956) is a greedy algorithm that is guaranteed to find a maximum weight spanning tree of a weighted graph. In the setting of density estimation, this procedure was proposed by Chow and Liu (1968) as a way of constructing a tree approximation to a distribution. At each stage the algorithm adds an edge connecting that pair of variables with maximum mutual information among all pairs not yet visited by the algorithm, if doing so does not form a cycle. When stopped early, after $k < d - 1$ edges have been added, it yields the best k -edge weighted forest.

Of course, the above procedure is not practical since the true density $p^*(x)$ is unknown. We replace the population mutual information $I(X_i; X_j)$ in (2.9) by the plug-in estimate $\hat{I}_n(X_i, X_j)$, defined as

$$\hat{I}_n(X_i, X_j) = \int_{\mathcal{X}_i \times \mathcal{X}_j} \hat{p}_n(x_i, x_j) \log \frac{\hat{p}_n(x_i, x_j)}{\hat{p}_n(x_i) \hat{p}_n(x_j)} dx_i dx_j \quad (3.2)$$

where $\hat{p}_n(x_i, x_j)$ and $\hat{p}_n(x_i)$ are bivariate and univariate kernel density estimates. Given this estimated mutual information matrix $\widehat{M}_n = [\hat{I}_n(X_i, X_j)]$, we can then apply Kruskal's algorithm (equivalently, the Chow-Liu algorithm) to find the best tree structure \hat{F}_n .

Since the number of edges of \hat{F}_n controls the number of degrees of freedom in the final density estimator, we need an automatic data-dependent way to choose it. We adopt the following two-stage procedure. First, randomly partition the data into two sets \mathcal{D}_1 and \mathcal{D}_2 of sizes n_1 and n_2 ; then, apply the following steps:

1. Using \mathcal{D}_1 , construct kernel density estimates of the univariate and bivariate marginals and calculate $\hat{I}_{n_1}(X_i, X_j)$ for $i, j \in \{1, \dots, d\}$ with $i \neq j$. Construct a full tree $\hat{F}_{n_1}^{(d-1)}$ with $d - 1$ edges, using the Chow-Liu algorithm.
2. Using \mathcal{D}_2 , prune the tree $\hat{F}_{n_1}^{(d-1)}$ to find a forest $\hat{F}_{n_1}^{(\hat{k})}$ with \hat{k} edges, for $0 \leq \hat{k} \leq d - 1$.

Once $\hat{F}_{n_1}^{(\hat{k})}$ is obtained in Step 2, we can calculate $\hat{p}_{\hat{F}_{n_1}^{(\hat{k})}}$ according to (2.1), using the kernel density estimates constructed in Step 1.

Algorithm 3.1 Chow-Liu (Kruskal)

- 1: **Input** data $\mathcal{D}_1 = \{X^{(1)}, \dots, X^{(n_1)}\}$.
 - 2: Calculate \widehat{M}_{n_1} , according to (3.3), (3.4), and (3.5).
 - 3: Initialize $E^{(0)} = \emptyset$
 - 4: **for** $k = 1, \dots, d - 1$ **do**
 - 5: $(i^{(k)}, j^{(k)}) \leftarrow \arg \max_{(i,j)} \widehat{M}_{n_1}(i, j)$ such that $E^{(k-1)} \cup \{(i^{(k)}, j^{(k)})\}$ does not contain a cycle
 - 6: $E^{(k)} \leftarrow E^{(k-1)} \cup \{(i^{(k)}, j^{(k)})\}$
 - 7: **Output** tree $\widehat{F}_{n_1}^{(d-1)}$ with edge set $E^{(d-1)}$.
-

3.1 Step 1: Estimating the marginals

Step 1 is carried out on the dataset \mathcal{D}_1 . Let $K(\cdot)$ be a univariate kernel function. Given an evaluation point (x_i, x_j) , the bivariate kernel density estimate for (X_i, X_j) based on the observations $\{X_i^{(s)}, X_j^{(s)}\}_{s \in \mathcal{D}_1}$ is defined as

$$\widehat{p}_{n_1}(x_i, x_j) = \frac{1}{n_1} \sum_{s \in \mathcal{D}_1} \frac{1}{h_2^2} K\left(\frac{X_i^{(s)} - x_i}{h_2}\right) K\left(\frac{X_j^{(s)} - x_j}{h_2}\right), \quad (3.3)$$

where we use a product kernel with $h_2 > 0$ as the bandwidth parameter. The univariate kernel density estimate $\widehat{p}_{n_1}(x_k)$ for X_k is

$$\widehat{p}_{n_1}(x_k) = \frac{1}{n_1} \sum_{s \in \mathcal{D}_1} \frac{1}{h_1} K\left(\frac{X_k^{(s)} - x_k}{h_1}\right), \quad (3.4)$$

where $h_1 > 0$ is the univariate bandwidth. Detailed specifications for $K(\cdot)$ and h_1, h_2 will be discussed in the next section.

We assume that the data lie in a d -dimensional unit cube $\mathcal{X} = [0, 1]^d$. To calculate the empirical mutual information $\widehat{I}_{n_1}(X_i, X_j)$, we need to numerically evaluate a two-dimensional integral. To do so, we calculate the kernel density estimates on a grid of points. We choose m evaluation points on each dimension, $x_{1i} < x_{2i} < \dots < x_{mi}$ for the i th variable. The mutual information $\widehat{I}_{n_1}(X_i, X_j)$ is then approximated as

$$\widehat{I}_{n_1}(X_i, X_j) = \frac{1}{m^2} \sum_{k=1}^m \sum_{\ell=1}^m \widehat{p}_{n_1}(x_{ki}, x_{\ell j}) \log \frac{\widehat{p}_{n_1}(x_{ki}, x_{\ell j})}{\widehat{p}_{n_1}(x_{ki}) \widehat{p}_{n_1}(x_{\ell j})}. \quad (3.5)$$

The approximation error can be made arbitrarily small by choosing m sufficiently large. As a practical concern, care needs to be taken that the factors $\widehat{p}_{n_1}(x_{ki})$ and $\widehat{p}_{n_1}(x_{\ell j})$ in the denominator are not too small; a truncation procedure can be used to ensure this. Once the $d \times d$ mutual information matrix $\widehat{M}_{n_1} = [\widehat{I}_{n_1}(X_i, X_j)]$ is obtained, we can apply the Chow-Liu (Kruskal) algorithm to find a maximum weight spanning tree.

3.2 Step 2: Optimizing the forest

The full tree $\widehat{F}_{n_1}^{(d-1)}$ obtained in Step 1 might have high variance when the dimension d is large, leading to over fitting in the density estimate. In order to reduce the variance, we prune the tree; that is, we choose forest with $k \leq d - 1$ edges. The number of edges k is a tuning parameter that induces a bias-variance tradeoff.

In order to choose k , note that in stage k of the Chow-Liu algorithm we have an edge set $E^{(k)}$ (in the notation of the Algorithm 3.1) which corresponds to a forest $\widehat{F}_{n_1}^{(k)}$ with k edges, where $\widehat{F}_{n_1}^{(0)}$ is the union of d disconnected nodes. To select k , we choose among the d trees $\widehat{F}_{n_1}^{(0)}, \widehat{F}_{n_1}^{(1)}, \dots, \widehat{F}_{n_1}^{(d-1)}$.

Let $\widehat{p}_{n_2}(x_i, x_j)$ and $\widehat{p}_{n_2}(x_k)$ be defined as in (3.3) and (3.4), but now evaluated solely based on the held-out data in \mathcal{D}_2 . For a density p_F that is supported by a forest F , we define the held-out negative log-likelihood risk as

$$\begin{aligned} \widehat{R}_{n_2}(p_F) &= - \sum_{(i,j) \in E_F} \int_{\mathcal{X}_i \times \mathcal{X}_j} \widehat{p}_{n_2}(x_i, x_j) \log \frac{p(x_i, x_j)}{p(x_i)p(x_j)} dx_i dx_j - \sum_{k \in V_F} \int_{\mathcal{X}_k} \widehat{p}_{n_2}(x_k) \log p(x_k) dx_k. \end{aligned} \quad (3.6)$$

The selected forest is then $\widehat{F}_{n_1}^{(\widehat{k})}$ where

$$\widehat{k} = \arg \min_{k \in \{0, \dots, d-1\}} \widehat{R}_{n_2} \left(\widehat{p}_{\widehat{F}_{n_1}^{(k)}} \right) \quad (3.7)$$

and where $\widehat{p}_{\widehat{F}_{n_1}^{(k)}}$ is computed using the density estimate \widehat{p}_{n_1} constructed on \mathcal{D}_1 .

For computational simplicity, we can also estimate \widehat{k} as

$$\widehat{k} = \arg \max_{k \in \{0, \dots, d-1\}} \frac{1}{n_2} \sum_{s \in \mathcal{D}_2} \log \left(\prod_{(i,j) \in E^{(k)}} \frac{\widehat{p}_{n_1}(X_i^{(s)}, X_j^{(s)})}{\widehat{p}_{n_1}(X_i^{(s)}) \widehat{p}_{n_1}(X_j^{(s)})} \prod_{k \in V_{\widehat{F}_{n_1}^{(k)}}} \widehat{p}_{n_1}(X_k^{(s)}) \right) \quad (3.8)$$

$$= \arg \max_{k \in \{0, \dots, d-1\}} \frac{1}{n_2} \sum_{s \in \mathcal{D}_2} \log \left(\prod_{(i,j) \in E^{(k)}} \frac{\widehat{p}_{n_1}(X_i^{(s)}, X_j^{(s)})}{\widehat{p}_{n_1}(X_i^{(s)}) \widehat{p}_{n_1}(X_j^{(s)})} \right). \quad (3.9)$$

This minimization can be efficiently carried out by iterating over the $d-1$ edges in $\widehat{F}_{n_1}^{(d-1)}$.

Once \widehat{k} is obtained, the final forest density estimate is given by

$$\widehat{p}_n(x) = \prod_{(i,j) \in E^{(\widehat{k})}} \frac{\widehat{p}_{n_1}(x_i, x_j)}{\widehat{p}_{n_1}(x_i) \widehat{p}_{n_1}(x_j)} \prod_k \widehat{p}_{n_1}(x_k). \quad (3.10)$$

4 Statistical Properties

In this section we present our theoretical results on risk consistency and structure selection consistency of the forest density estimate $\widehat{p}_n = \widehat{p}_{\widehat{F}_d^{(\widehat{k})}}$.

To establish some notation, we write $a_n = \Omega(b_n)$ if there exists a constant c such that $a_n \geq cb_n$ for sufficiently large n . We also write $a_n \asymp b_n$ if there exists a constant c such that $a_n \leq cb_n$ and $b_n \leq ca_n$ for sufficiently large n . Given a d -dimensional function f on the domain \mathcal{X} , we denote its $L_2(P)$ -norm and sup-norm as

$$\|f\|_{L_2(P)} = \sqrt{\int_{\mathcal{X}} f^2(x) dP_X(x)}, \quad \|f\|_{\infty} = \sup_{x \in \mathcal{X}} |f(x)| \quad (4.1)$$

where P_X is the probability measure induced by X . Throughout this section, all constants are treated as generic values, and as a result they can change from line to line.

In our use of a data splitting scheme, we always adopt equally sized splits for simplicity, so that $n_1 = n_2 = n/2$, noting that this does not affect the final rate of convergence.

4.1 Assumptions on the density

Fix $\beta > 0$. For any d -tuple $\alpha = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}^d$ and $x = (x_1, \dots, x_d) \in \mathcal{X}$, we define $x^\alpha = \prod_{j=1}^d x_j^{\alpha_j}$. Let D^α denote the differential operator

$$D^\alpha = \frac{\partial^{\alpha_1 + \dots + \alpha_d}}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}. \quad (4.2)$$

For any real-valued d -dimensional function f on \mathcal{X} that is $\lfloor \beta \rfloor$ -times continuously differentiable at point $x_0 \in \mathcal{X}$, let $P_{f, x_0}^{(\beta)}(x)$ be its Taylor polynomial of degree $\lfloor \beta \rfloor$ at point x_0 :

$$P_{f, x_0}^{(\beta)}(x) = \sum_{\alpha_1 + \dots + \alpha_d \leq \lfloor \beta \rfloor} \frac{(x - x_0)^\alpha}{\alpha_1! \dots \alpha_d!} D^\alpha f(x_0). \quad (4.3)$$

Fix $L > 0$, and denote by $\Sigma(\beta, L, r, x_0)$ the set of functions $f : \mathcal{X} \rightarrow \mathbb{R}$ that are $\lfloor \beta \rfloor$ -times continuously differentiable at x_0 and satisfy

$$\left| f(x) - P_{f, x_0}^{(\beta)}(x) \right| \leq L \|x - x_0\|_2^\beta, \quad \forall x \in \mathcal{B}(x_0, r) \quad (4.4)$$

where $\mathcal{B}(x_0, r) = \{x : \|x - x_0\|_2 \leq r\}$ is the L_2 -ball of radius r centered at x_0 . The set $\Sigma(\beta, L, r, x_0)$ is called the (β, L, r, x_0) -locally Hölder class of functions. Given a set A , we define

$$\Sigma(\beta, L, r, A) = \bigcap_{x_0 \in A} \Sigma(\beta, L, r, x_0). \quad (4.5)$$

The following are the regularity assumptions we make on the true density function $p^*(x)$.

Assumption 4.1 For any $1 \leq i < j \leq d$, we assume

(D1) there exist $L_1 > 0$ and $L_2 > 0$ such that for any $c > 0$ the true bivariate and univariate densities satisfy

$$p^*(x_i, x_j) \in \Sigma \left(\beta, L_2, c (\log n/n)^{\frac{1}{2\beta+2}}, \mathcal{X}_i \times \mathcal{X}_j \right) \quad (4.6)$$

and

$$p^*(x_i) \in \Sigma \left(\beta, L_1, c (\log n/n)^{\frac{1}{2\beta+1}}, \mathcal{X}_i \right); \quad (4.7)$$

(D2) there exists two constants c_1 and c_2 such that

$$c_1 \gamma_n \leq \inf_{x_i, x_j \in \mathcal{X}_i \times \mathcal{X}_j} p^*(x_i, x_j) \leq \sup_{x_i, x_j \in \mathcal{X}_i \times \mathcal{X}_j} p^*(x_i, x_j) \leq c_2 \quad (4.8)$$

$$\mu\text{-almost surely, where } \gamma_n^2 = \Omega \left(\sqrt{\frac{\log n + \log d}{n^{\beta/(\beta+1)}}} \right).$$

These assumptions are mild, in the sense that instead of adding constraints on the joint density $p^*(x)$, we only add regularity conditions on the bivariate and univariate marginals.

4.2 Assumptions on the kernel

An important ingredient in our analysis is an exponential concentration result for the kernel density estimate, due to Giné and Guillou (2002). We first specify the requirements on the kernel function $K(\cdot)$.

Let (Ω, \mathcal{A}) be a measurable space and let \mathcal{F} be a uniformly bounded collection of measurable functions.

Definition 4.2 \mathcal{F} is a bounded measurable VC class of functions with characteristics A and v if it is separable and for every probability measure P on (Ω, \mathcal{A}) and any $0 < \epsilon < 1$,

$$N(\epsilon \|F\|_{L_2(P)}, \mathcal{F}, \|\cdot\|_{L_2(P)}) \leq \left(\frac{A}{\epsilon} \right)^v, \quad (4.9)$$

where $F(x) = \sup_{f \in \mathcal{F}} |f(x)|$ and $N(\epsilon, \mathcal{F}, \|\cdot\|_{L_2(P)})$ denotes the ϵ -covering number of the metric space $(\Omega, \|\cdot\|_{L_2(P)})$; that is, the smallest number of balls of radius no larger than ϵ (in the norm $\|\cdot\|_{L_2(P)}$) needed to cover \mathcal{F} .

The one-dimensional density estimates are constructed using a kernel K , and the two-dimensional estimates are constructed using the product kernel

$$K_2(x, y) = K(x) \cdot K(y). \quad (4.10)$$

Assumption 4.3 The kernel K satisfies the following properties.

(K1) $\int_{-\infty}^{\infty} K(u) du = 1$, $\int_{-\infty}^{\infty} K^2(u) du < \infty$ and $\sup_{u \in \mathbb{R}} K(u) \leq c$ for some constant c .

(K2) K is a finite linear combination of functions g whose epigraphs $\text{epi}(g) = \{(s, u) : g(s) \geq u\}$, can be represented as a finite number of Boolean operations (union and intersection) among sets of the form $\{(s, u) : Q(s, u) \geq \phi(u)\}$, where Q is a polynomial on $\mathbb{R} \times \mathbb{R}$ and ϕ is an arbitrary real function.

(K3) K has a compact support and for any $\ell \geq 1$ and $1 \leq \ell' \leq \lfloor \beta \rfloor$

$$\int |t|^\beta |K(t)| dt < \infty, \text{ and } \int |K(t)|^\ell dt < \infty, \int t^{\ell'} K(t) dt = 0. \quad (4.11)$$

Assumptions (K1), (K2) and (K3) are mild. As pointed out by Nolan and Pollard (1987), both the pyramid (truncated or not) kernel and the boxcar kernel satisfy them. It follows from (K2) that the classes of functions

$$\mathcal{F}_1 = \left\{ \frac{1}{h_1} K \left(\frac{u - \cdot}{h_1} \right) : u \in \mathbb{R}, h_1 > 0 \right\} \quad (4.12)$$

$$\mathcal{F}_2 = \left\{ \frac{1}{h_2^2} K \left(\frac{u - \cdot}{h_2} \right) K \left(\frac{t - \cdot}{h_2} \right) : u, t \in \mathbb{R}, h_2 > 0 \right\} \quad (4.13)$$

are bounded VC classes, in the sense of Definition 4.2. Assumption (K3) essentially says that the kernel $K(\cdot)$ should be β -valid; see Tsybakov (2008) and Definition 6.1 in Rigollet and Vert (2009) for further details about this assumption.

We choose the bandwidths h_1 and h_2 used in the one-dimensional and two-dimensional kernel density estimates to satisfy

$$h_1 \asymp \left(\frac{\log n}{n} \right)^{\frac{1}{1+2\beta}} \quad (4.14)$$

$$h_2 \asymp \left(\frac{\log n}{n} \right)^{\frac{1}{2+2\beta}}. \quad (4.15)$$

This choice of bandwidths ensures the optimal rate of convergence.

4.3 Risk consistency

Given the above assumptions, we first present a key lemma that establishes the rates of convergence of bivariate and univariate kernel density estimates in the sup norm. Due to space limitations, the proof of this and our other technical results are provided in the extended arXiv version of this paper (Liu et al., 2010).

Lemma 4.4 *Under Assumptions 4.1 and 4.3, and choosing bandwidths satisfying (4.14) and (4.15), the bivariate and univariate kernel density estimates $\widehat{p}(x_i, x_j)$ and $\widehat{p}(x_k)$ in (3.3) and (3.4) satisfy*

$$\max_{(i,j) \in \{1, \dots, d\} \times \{1, \dots, d\}} \sup_{(x_i, x_j) \in \mathcal{X}_i \times \mathcal{X}_j} |\widehat{p}(x_i, x_j) - p^*(x_i, x_j)| = O_P \left(\sqrt{\frac{\log n + \log d}{n^{\beta/(1+2\beta)}}} \right) \quad (4.16)$$

and

$$\max_{k \in \{1, \dots, d\}} \sup_{x_k \in \mathcal{X}_k} |\widehat{p}(x_k) - p^*(x_k)| = O_P \left(\sqrt{\frac{\log n + \log d}{n^{2\beta/(1+2\beta)}}} \right). \quad (4.17)$$

To describe the risk consistency result, let $\mathcal{P}_d^{(d-1)} = \mathcal{P}_d$ be the family of densities that are supported by forests with at most $d-1$ edges, as already defined in (2.2). For $0 \leq k \leq d-1$, we define $\mathcal{P}_d^{(k)}$ as the family of d -dimensional densities that are supported by forests with at most k edges. Then

$$\mathcal{P}_d^{(0)} \subset \mathcal{P}_d^{(1)} \subset \dots \subset \mathcal{P}_d^{(d-1)}. \quad (4.18)$$

Now, due to the nesting property (4.18), we have

$$\inf_{q_F \in \mathcal{P}_d^{(0)}} R(q_F) \geq \inf_{q_F \in \mathcal{P}_d^{(1)}} R(q_F) \geq \dots \geq \inf_{q_F \in \mathcal{P}_d^{(d-1)}} R(q_F). \quad (4.19)$$

We first analyze the forest density estimator obtained using a fixed number of edges $k < d$; specifically, consider stopping the Chow-Liu algorithm in Stage 1 after k iterations. This is in contrast to the algorithm described in 3.2, where the pruned tree size is automatically determined on the held out data. While this is not very realistic in applications, since the tuning parameter k is generally hard to choose, the analysis in this case is simpler, and can be directly exploited to analyze the more complicated data-dependent method.

Theorem 4.5 (Risk consistency) *Let $\widehat{p}_{\widehat{F}_d^{(k)}}$ be the forest density estimate with $|E(\widehat{F}_d^{(k)})| = k$, obtained after the first k iterations of the Chow-Liu algorithm, for some $k \in \{0, \dots, d-1\}$. Under Assumptions 4.1 and 4.3, we have*

$$R(\widehat{p}_{\widehat{F}_d^{(k)}}) - \inf_{q_F \in \mathcal{P}_d^{(k)}} R(q_F) = O_P \left(k \sqrt{\frac{\log n + \log d}{n^{\beta/(1+2\beta)}}} + d \sqrt{\frac{\log n + \log d}{n^{2\beta/(1+2\beta)}}} \right). \quad (4.20)$$

Note that this result allows the dimension d to increase at a rate $o\left(\sqrt{n^{2\beta/(1+2\beta)}/\log n}\right)$ and the number of edges k to increase at a rate $o\left(\sqrt{n^{\beta/(1+2\beta)}/\log n}\right)$, with the excess risk still decreasing to zero asymptotically.

The above results can be used to prove a risk consistency result for the data-dependent pruning method using the data-splitting scheme described in Section 3.2.

Theorem 4.6 *Let $\widehat{p}_{\widehat{F}_d^{(\widehat{k})}}$ be the forest density estimate using the data-dependent pruning method in Section 3.2, and let $\widehat{p}_{\widehat{F}_d^{(k)}}$ be the estimate with $|E(\widehat{F}_d^{(k)})| = k$ obtained after the first k iterations of the Chow-Liu algorithm. Under Assumptions 4.1 and 4.3, we have*

$$R(\widehat{p}_{\widehat{F}_d^{(\widehat{k})}}) - \min_{0 \leq k \leq d-1} R(\widehat{p}_{\widehat{F}_d^{(k)}}) = O_P \left((k^* + \widehat{k}) \sqrt{\frac{\log n + \log d}{n^{\beta/(1+2\beta)}}} + d \sqrt{\frac{\log n + \log d}{n^{2\beta/(1+2\beta)}}} \right) \quad (4.21)$$

where $k^* = \arg \min_{0 \leq k \leq d-1} R(\widehat{p}_{\widehat{F}_d^{(k)}})$.

The proof of this theorem is given in (Liu et al., 2010).

Algorithm 5.1 Approximate Max Weight t -Restricted Forest

- 1: **Input** graph G with positive edge weights, and positive integer $t \geq 2$.
 - 2: Sort edges in decreasing order of weight.
 - 3: Greedily add edges in decreasing order of weight such that
 - (a) the degree of any node is at most t ;
 - (b) no cycles are formed.The resulting forest is $F' = \{T_1, T_2, \dots, T_m\}$.
 - 4: **Output** $F_t = \cup_j \text{TreePartition}(T_j, t)$.
-

5 Tree Restricted Forests

We now turn to the problem of estimating forests with restricted tree sizes. As discussed in the introduction, clustering problems motivate the goal of constructing forest structured density estimators where each connected component has a restricted number of edges. But estimating restricted tree size forests can also be useful in model selection for the purpose of risk minimization, since the maximum subtree size can be viewed as an additional complexity parameter.

Definition 5.1 A t -restricted forest of a graph G is a subgraph F_t such that

1. F_t is the disjoint union of connected components $\{T_1, \dots, T_m\}$, each of which is a tree;
2. $|T_i| \leq t$ for each $i \leq m$, where $|T_i|$ denotes the number of edges in the i th component.

Given a weight w_e assigned to each edge of G , an optimal t -restricted forest F_t^* satisfies

$$w(F_t^*) \geq \max_{F_t(G)} w(F_t) \quad (5.1)$$

where $w(F) = \sum_{e \in F} w_e$ is the weight of a forest F and $\mathcal{F}_t(G)$ denotes the collection of all t -restricted forests of G .

For $t = 1$, the problem is maximum weighted matching. Unfortunately for $t \geq 2$, determining a maximum weight t -restricted forest is an NP-hard problem; however, this problem appears not to have been previously studied. Our reduction is from Exact 3-Cover (X3C), shown to be NP-complete by Garey and Johnson (1979)). In X3C, we are given a set X , a family \mathcal{S} of 3-element subsets of X , and we must choose a subfamily of disjoint 3-element subsets to cover X .

Our reduction constructs a graph with special tree-shaped subgraphs called *gadgets*, such that each gadget corresponds to a 3-element subset in \mathcal{S} . We show that finding a maximum weight t -restricted forest on this graph would allow us to then recover a solution to X3C by analyzing how the optimal forest must partition each of the gadgets.

Given the difficulty of finding an optimal t -restricted forest, it is of interest to study approximation algorithms. Algorithm 5.1 gives a procedure that has two stages. In the first stage, a forest is greedily constructed in such a way that each node has degree no larger than $t + 1$. In the second stage, each tree in the forest is partitioned in an optimal way by removing edges, resulting in a collection of trees, each of which has size at most t . The second stage employs a procedure we call `TreePartition` that takes a tree and returns the optimal t -restricted subforest. `TreePartition` is a divide-and-conquer procedure of Lukes (1974) that finds a carefully chosen set of forest partitions for each child subtree. It then merges these sets with the parent node one subtree at a time. The details of the `TreePartition` procedure are given in (Liu et al., 2010).

Theorem 5.2 Let F_t be the output of Algorithm 5.1, and let F_t^* be the optimal t -restricted forest. Then $w(F_t) \geq \frac{1}{4}w(F_t^*)$.

5.1 Pruning Based on t -Restricted Forests

For a given t , after producing an approximate maximum weight t -restricted forest \hat{F}_t using \mathcal{D}_1 , we prune away edges using \mathcal{D}_2 . To do so, we first construct a new set of univariate and bivariate kernel density estimates using \mathcal{D}_2 , as before, $\hat{p}_{n_2}(x_i)$ and $\hat{p}_{n_2}(x_i, x_j)$. We then estimate the “cross-entropies” of the kernel density estimates \hat{p}_{n_1} for each pair of variables by computing

$$\hat{I}_{n_2, n_1}(X_i, X_j) = \int \hat{p}_{n_2}(x_i, x_j) \log \frac{\hat{p}_{n_1}(x_i, x_j)}{\hat{p}_{n_1}(x_i)\hat{p}_{n_1}(x_j)} dx_i dx_j \quad (5.2)$$

$$\approx \frac{1}{m^2} \sum_{k=1}^m \sum_{\ell=1}^m \hat{p}_{n_2}(x_{k_i}, x_{\ell_j}) \log \frac{\hat{p}_{n_1}(x_{k_i}, x_{\ell_j})}{\hat{p}_{n_1}(x_{k_i})\hat{p}_{n_1}(x_{\ell_j})}. \quad (5.3)$$

Algorithm 5.2 t -Restricted Forest Density Estimation

- 1: Divide data into two halves \mathcal{D}_1 and \mathcal{D}_2 .
 - 2: Compute kernel density estimators \hat{p}_{n_1} and \hat{p}_{n_2} for all pairs and single variable marginals.
 - 3: For all pairs (i, j) compute $\hat{I}_{n_1}(X_i, X_j)$ according to (3.5) and $\hat{I}_{n_2, n_1}(X_i, X_j)$ according to (5.3).
 - 4: For $t = 0, \dots, t_{\text{final}}$ where t_{final} is chosen based on the application
 1. Compute or approximate (for $t \geq 2$) the optimal t -restricted forest \hat{F}_t using \hat{I}_{n_1} as edge weights.
 2. Prune \hat{F}_t to eliminate all edges with negative weights \hat{I}_{n_2, n_1} .
 - 5: Among all pruned forests \hat{p}_{F^t} , select $\hat{t} = \arg \min_{0 \leq t \leq t_{\text{final}}} \hat{R}_{n_2}(\hat{p}_{\hat{F}_t})$.
-

We then eliminate all edges (i, j) in \hat{F}_t for which $\hat{I}_{n_2, n_1}(X_i, X_j) \leq 0$. For notational simplicity, we denote the resulting pruned forest again by \hat{F}_t .

To estimate the risk, we simply use $\hat{R}_{n_2}(\hat{p}_{\hat{F}_t})$ as defined before, and select the forest $\hat{F}_{\hat{t}}$ according to

$$\hat{t} = \arg \min_{0 \leq t \leq d-1} \hat{R}_{n_2}(\hat{p}_{\hat{F}_t}). \quad (5.4)$$

The resulting procedure is summarized in Algorithm 5.2.

Using the approximation guarantee and our previous analysis, we have that the population weights of the approximate t -restricted forest and the optimal forest satisfy the following inequality. We state the result for a general c -approximation algorithm; for the algorithm given above, $c = 4$, but tighter approximations are possible.

Theorem 5.3 *Assume the conditions of Theorem 4.5. For $t \geq 2$, let \hat{F}_t be the forest constructed using a c -approximation algorithm, and let F_t^* be the optimal forest; both constructed with respect to finite sample edge weights $\hat{w}_{n_1} = \hat{I}_{n_1}$. Then*

$$w(\hat{F}_t) \geq \frac{1}{c} w(F_t^*) + O_P \left((k^* + \hat{k}) \sqrt{\frac{\log n + \log d}{n^{\beta/(1+\beta)}}} \right) \quad (5.5)$$

where \hat{k} and k^* are the number of edges in \hat{F}_t and F_t^* , respectively, and w denotes the population weights, given by the mutual information.

As seen below, although the approximation algorithm has weaker theoretical guarantees, it out-performs other approaches in experiments.

6 Experimental Results

In this section, we report numerical results on both synthetic datasets and microarray data; additional experiments and further details are presented in the extended version of this paper (Liu et al., 2010). We mainly compare the forest density estimator with sparse Gaussian graphical models, fitting a multivariate Gaussian with a sparse inverse covariance matrix. The sparse Gaussian models are estimated using the graphical lasso algorithm (glasso) of Friedman et al. (2007), which is a refined version of an algorithm first derived by Banerjee et al. (2008). Since the glasso typically results in a large parameter bias as a consequence of the ℓ_1 regularization, we also compare with a method that we call the *refit glasso*, which is a two-step procedure—in the first step, a sparse inverse covariance matrix is obtained by the glasso; in the second step, a Gaussian model is refit without ℓ_1 regularization, but enforcing the sparsity pattern obtained in the first step.

6.1 Synthetic data

We generate high dimensional Gaussian and non-Gaussian data which are consistent with an undirected graph. A typical run showing the held-out log-likelihood and estimated graphs is provided in Figure 6.1. We see that for the Gaussian data, the refit glasso has a higher held-out log-likelihood than the forest density estimator and the glasso. This is expected, since the Gaussian model is correct. For very sparse models, however, the performance of the glasso is worse than that of the forest density estimator, due to the large parameter bias resulting from the ℓ_1 regularization. We also observe an efficiency loss in the nonparametric forest density estimator, compared to the refit glasso. The graphs are automatically selected using the held-out log-likelihood, and we see that the nonparametric forest-based kernel density estimator tends to select a sparser model, while the parametric Gaussian models tend to overselect.

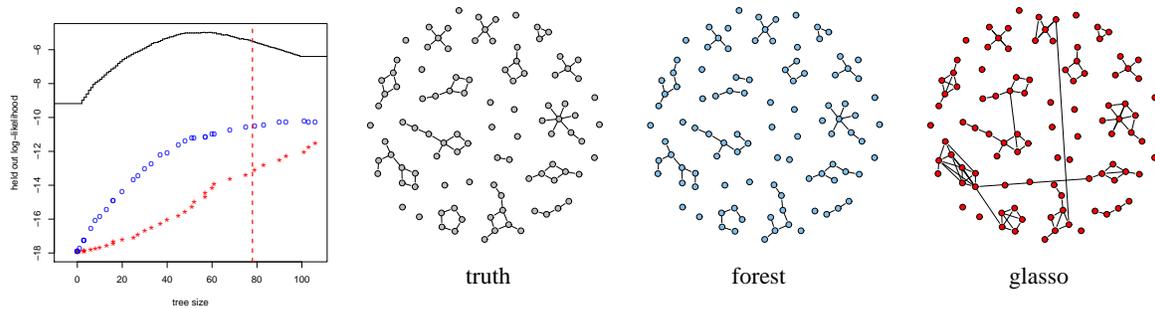


Figure 6.1: Synthetic data, non-Gaussian. Held-out log-likelihood plots show forest density (black step function), glasso (red stars), and refit glasso (blue circles); vertical indicates size of true graph.

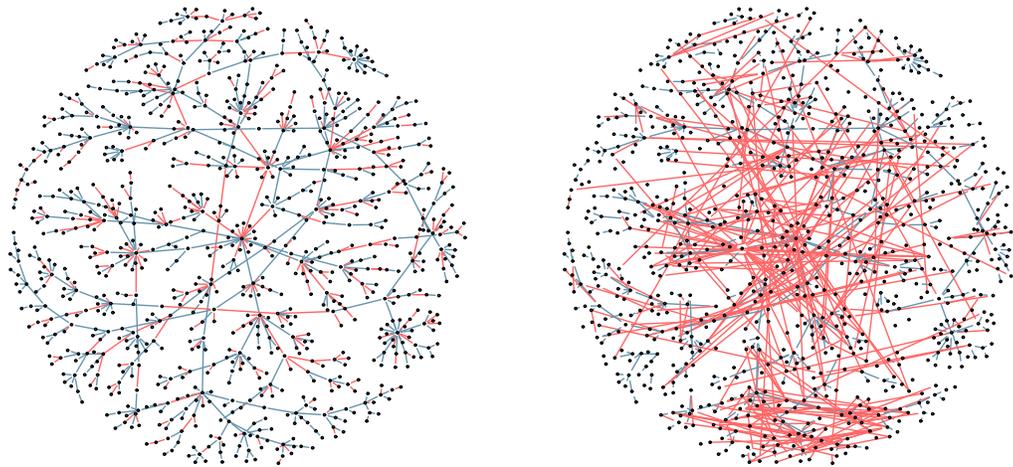


Figure 6.2: A 934 gene subgraph of the full estimated 4238 gene network. Left: estimated forest graph. Right: estimated Gaussian graph. Red edges in the forest graph are missing from the Gaussian graph and vice versa; the blue edges are shared by both graphs. Note that the layout of the genes is the same for both graphs.

6.2 Microarray Data

Our data comes from Nayak et al. (2009). The dataset contains Affymetrix chip measured expression levels of 4238 genes for 295 normal subjects in the *Centre d'Etude du Polymorphisme Humain* (CEPH) and the International HapMap collections. The 295 subjects come from four different groups: 148 unrelated grandparents in the CEPH-Utah pedigrees, 43 Han Chinese in Beijing, 44 Japanese in Tokyo, and 60 Yoruba in Ibadan, Nigeria. Since we want to find common network patterns across different groups of subjects, we pooled the data together into a $n = 295$ by $p = 4238$ numerical matrix.

We estimate the full 4238 node graph using both the forest density estimator (described in Section 3.1 and 3.2) and the Meinshausen-Bühlmann neighborhood search method (Meinshausen & Bühlmann, 2006) with regularization parameter chosen to give it about same number as edges as the forest graph. The forest density estimated graph reveals one strongly connected component of more than 3000 genes and various isolated genes; this is consistent with the analysis in Nayak et al. (2009) and is realistic for the regulatory system of humans. The Gaussian graph contains similar component structure, but the set of edges differs significantly. We also ran the t -restricted forest algorithm for $t = 2000$ and it successfully separates the giant component into three smaller components. Since the forest density estimator produces a sparse and interpretable graph whose structure is consistent with biological analysis, we believe that it may be helpful for studying gene interaction networks.

For visualization purposes, we show only a 934 gene subgraph of the strongly connected component among the full 4238 node graphs we estimated. We refer the reader to the extended arXiv version of this paper (Liu et al., 2010) for the full graph and other visualizations.

7 Conclusion

We have studied forest density estimation for high dimensional data. Forest density estimation skirts the curse of dimensionality by restricting to undirected graphs without cycles, while allowing fully nonparametric marginal densities. The method is computationally simple, and the optimal size of the forest can be robustly selected by a data-splitting scheme. We have established oracle properties and rates of convergence for function estimation in this setting. Our experimental results compared the forest density estimator to the sparse Gaussian graphical model in terms of both predictive risk and the qualitative properties of the estimated graphs for human gene expression array data. Together, these results indicate that forest density estimation can be a useful tool for relaxing the normality assumption in graphical modeling.

Acknowledgements

The research reported here was supported in part by NSF grant CCF-0625879, AFOSR contract FA9550-09-1-0373, and a grant from Google. We thank Haijie Gu for assistance in running the human gene expression data experiments.

References

- Aigner, M., & Ziegler, G. (1998). *Proofs from THE BOOK*. Springer-Verlag.
- Bach, F. R., & Jordan, M. I. (2003). Beyond independent components: Trees and clusters. *Journal of Machine Learning Research*, 4, 1205–1233.
- Banerjee, O., El Ghaoui, L., & d’Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation. *Journal of Machine Learning Research*, 9, 485–516.
- Cayley, A. (1889). A theorem on trees. *Quart. J. Math.*, 23, 376–378.
- Checheta, A., & Guestrin, C. (2007). Efficient principled learning of thin junction trees. *In Advances in Neural Information Processing Systems (NIPS)*. Vancouver, Canada.
- Chow, C., & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14, 462–467.
- Friedman, J. H., Hastie, T., & Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9, 432–441.
- Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of np-completeness*. W. H. Freeman.
- Giné, E., & Guillaou, A. (2002). Rates of strong uniform consistency for multivariate kernel density estimators. *Annales de l’institut Henri Poincaré (B), Probabilités et Statistiques*, 38, 907–921.
- Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7, 48–50.
- Lauritzen, S. L. (1996). *Graphical models*. Clarendon Press.
- Liu, H., Xu, M., Gu, H., Gupta, A., Lafferty, J., & Wasserman, L. (2010). Forest density estimation. arXiv:1001.1557.
- Lukes, J. A. (1974). Efficient algorithm for the partitioning of trees. *IBM Jour. of Res. and Dev.*, 18, 274.
- Meinshausen, N., & Bühlmann, P. (2006). High dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, 34, 1436–1462.
- Nayak, R., Kearns, M., Spielman, R., & Cheung, V. (2009). Coexpression network based on natural variation in human gene expression reveals gene interactions and functions. *Genome Research*, 19, 1953–1962.
- Nolan, D., & Pollard, D. (1987). U-processes: Rates of convergence. *The Annals of Statistics*, 15, 780 – 799.
- Rigollet, P., & Vert, R. (2009). Fast rates for plug-in estimators of density level sets. *Bernoulli (to appear)*.
- Tan, V., Anandkumar, A., Tong, L., & Willsky, A. (2009a). A large-deviation analysis for the maximum likelihood learning of tree structures. arXiv:0905.0940.

- Tan, V., Anandkumar, A., & Willsky, A. (2009b). Learning Gaussian tree models: Analysis of error exponents and extremal structures. arXiv:0909.5216.
- Tsybakov, A. B. (2008). *Introduction to nonparametric estimation*. Springer Publishing Company, Incorporated.
- Wille, A., Zimmermann, P., Vranová, E., Fürholz, A., Laule, O., Bleuler, S., Hennig, L., Prelić, A., von Rohr, P., Thiele, L., Zitzler, E., Gruissem, W., & Bühlmann, P. (2004). Sparse Gaussian graphical modelling of the isoprenoid gene network in *Arabidopsis thaliana*. *Genome Biology*, 5, R92.

Toward Learning Gaussian Mixtures with Arbitrary Separation

Mikhail Belkin
Ohio State University
Columbus, Ohio
mbelkin@cse.ohio-state.edu

Kaushik Sinha
Ohio State University
Columbus, Ohio
sinhak@cse.ohio-state.edu

Abstract

In recent years analysis of complexity of learning Gaussian mixture models from sampled data has received significant attention in computational machine learning and theory communities. In this paper we present the first result showing that polynomial time learning of multidimensional Gaussian Mixture distributions is possible when the separation between the component means is arbitrarily small. Specifically, we present an algorithm for learning the parameters of a mixture of k identical spherical Gaussians in n -dimensional space with an arbitrarily small separation between the components, which is polynomial in dimension, inverse component separation and other input parameters for a fixed number of components k . The algorithm uses a projection to k dimensions and then a reduction to the 1-dimensional case. It relies on a theoretical analysis showing that two 1-dimensional mixtures whose densities are close in the L^2 norm must have similar means and mixing coefficients. To produce the necessary lower bound for the L^2 norm in terms of the distances between the corresponding means, we analyze the behavior of the Fourier transform of a mixture of Gaussians in one dimension around the origin, which turns out to be closely related to the properties of the Vandermonde matrix obtained from the component means. Analysis of minors of the Vandermonde matrix together with basic function approximation results allows us to provide a lower bound for the norm of the mixture in the Fourier domain and hence a bound in the original space. Additionally, we present a separate argument for reconstructing variance.

1 Introduction

Mixture models, particularly Gaussian mixture models, are a widely used tool for many problems of statistical inference (Titterton et al., 1985; McLachlan & Peel, 2000; McLachlan & Basford, 1988; Everitt & Hand, 1981; Lindsay, 1995). The basic problem is to estimate the parameters of a mixture distribution, such as the mixing coefficients, means and variances within some pre-specified precision from a number of sampled data points. While the history of Gaussian mixture models goes back to (Pearson, 1894), in recent years the theoretical aspects of mixture learning have attracted considerable attention in the theoretical computer science, starting with the pioneering work of (Dasgupta, 1999), who showed that a mixture of k spherical Gaussians in n dimensions can be learned in time polynomial in n , provided certain separation conditions between the component means (separation of order \sqrt{n}) are satisfied. This work has been refined and extended in a number of recent papers. The first result from (Dasgupta, 1999) was later improved to the order of $\Omega(n^{\frac{1}{4}})$ in (Dasgupta & Schulman, 2000) for spherical Gaussians and in (Arora & Kannan, 2001) for general Gaussians. The separation requirement was further reduced and made independent of n to the order of $\Omega(k^{\frac{1}{4}})$ in (Vempala & Wang, 2002) for spherical Gaussians and to the order of $\Omega(\frac{k^{\frac{3}{2}}}{\epsilon^2})$ in (Kannan et al., 2005) for Logconcave distributions. In a related work (Achlioptas & McSherry, 2005) the separation requirement was reduced to $\Omega(k + \sqrt{k \log n})$. An extension of PCA called isotropic PCA was introduced in (Brubaker & Vempala, 2008) to learn mixtures of Gaussians when any pair of Gaussian components is separated by a hyperplane having very small overlap along the hyperplane direction (so-called "pancake layering problem").

In a slightly different direction the recent work (Feldman et al., 2006) made an important contribution to the subject by providing a polynomial time algorithm for PAC-style learning of mixture of Gaussian distributions with arbitrary separation between the means. The authors used a grid search over the space of parameters to a construct a hypothesis mixture of Gaussians that has density close to the actual mixture generating the data. We note that the problem analyzed in (Feldman et al., 2006) can be viewed as density estimation within

a certain family of distributions and is different from most other work on the subject, including our paper, which address parameter learning¹.

We also note several recent papers dealing with the related problems of learning mixture of product distributions and heavy tailed distributions. See for example, (Feldman et al., 2008; Dasgupta et al., 2005; Chaudhuri & Rao, 2008a; Chaudhuri & Rao, 2008b).

In the statistics literature, (Chen, 1995) showed that optimal convergence rate of MLE estimator for finite mixture of normal distributions is $O(n^{-\frac{1}{2}})$, where n is the sample size, if number of mixing components k is known in advance and is $O(n^{-\frac{1}{4}})$ when the number of mixing components is known up to an upper bound. However, this result does not address the computational aspects, especially in high dimension.

In this paper we develop a polynomial time (for a fixed k) algorithm to identify the parameters of the mixture of k identical spherical Gaussians with potentially unknown variance for an arbitrarily small separation between the components². To the best of our knowledge this is the first result of this kind except for the simultaneous and independent work (Kalai et al., 2010), which analyzes the case of a mixture of two Gaussians with arbitrary covariance matrices using the method of moments. We note that the results in (Kalai et al., 2010) and in our paper are somewhat orthogonal. Each paper deals with a special case of the ultimate goal (two arbitrary Gaussians in (Kalai et al., 2010) and k identical spherical Gaussians with unknown variance in our case), which is to show polynomial learnability for a mixture with an arbitrary number of components and arbitrary variance.

All other existing algorithms for parameter estimation require minimum separation between the components to be an increasing function of at least one of n or k . Our result also implies a density estimate bound along the lines of (Feldman et al., 2006). We note, however, that we do have to pay a price as our procedure (similarly to that in (Feldman et al., 2006)) is super-exponential in k . Despite these limitations we believe that our paper makes a step towards understanding the fundamental problem of polynomial learnability of Gaussian mixture distributions. We also think that the technique used in the paper to obtain the lower bound may be of independent interest.

The main algorithm in our paper involves a grid search over a certain space of parameters, specifically means and mixing coefficients of the mixture (a completely separate argument is given to estimate the variance). By giving appropriate lower and upper bounds for the norm of the difference of two mixture distributions in terms of their means, we show that such a grid search is guaranteed to find a mixture with nearly correct values of the parameters.

To prove that, we need to provide a lower and upper bounds on the norm of the mixture. A key point of our paper is the lower bound showing that two mixtures with different means cannot produce similar density functions. This bound is obtained by reducing the problem to a 1-dimensional mixture distribution and analyzing the behavior of the Fourier transform (closely related to the characteristic function, whose coefficients are moments of a random variable up to multiplication by a power of the imaginary unit i) of the difference between densities near zero. We use certain properties of minors of Vandermonde matrices to show that the norm of the mixture in the Fourier domain is bounded from below. Since the L^2 norm is invariant under the Fourier transform this provides a lower bound on the norm of the mixture in the original space.

We also note the work (Lindsay, 1989), where Vandermonde matrices appear in the analysis of mixture distributions in the context of proving consistency of the method of moments (in fact, we rely on a result from (Lindsay, 1989) to provide an estimate for the variance).

Finally, our lower bound, together with an upper bound and some results from the non-parametric density estimation and spectral projections of mixture distributions allows us to set up a grid search algorithm over the space of parameters with the desired guarantees.

Extended version of this paper is available at <http://arxiv.org/abs/0907.1054>.

2 Outline of the argument

In this section we provide an informal outline of the argument that leads to the main result. To simplify the discussion, we will assume that the variance for the components is known or estimated by using the estimation algorithm provided in Section 3.3. It is straightforward (but requires a lot of technical details) to see that all results go through if the actual variance is replaced by a sufficiently (polynomially) accurate estimate.

We will denote the n -dimensional Gaussian density $\frac{1}{(\sqrt{2\pi}\sigma)^n} \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\sigma^2}\right)$ by $K(\mathbf{x}, \boldsymbol{\mu})$, where $\mathbf{x}, \boldsymbol{\mu} \in$

¹Note that density estimation is generally easier than parameter learning since quite different configurations of parameters could conceivably lead to very similar density functions, while similar configurations of parameters always result in similar density functions.

²We point out that some non-zero separation is necessary since the problem of learning parameters without any separation assumptions at all is ill-defined.

\mathbb{R}^n or, when appropriate, in \mathbb{R}^k . The notation $\|\cdot\|$ will always be used to represent L^2 norm while $d_H(\cdot, \cdot)$ will be used to denote the Hausdorff distance between sets of points. Let $p(\mathbf{x}) = \sum_{i=1}^k \alpha_i K(\mathbf{x}, \boldsymbol{\mu}_i)$ be a mixture of k Gaussian components with the covariance matrix $\sigma^2 I$ in \mathbb{R}^n . The goal will be to identify the means $\boldsymbol{\mu}_i$ and the mixing coefficients α_i under the assumption that the minimum distance $\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|, i \neq j$ is bounded from below by some given (arbitrarily small) d_{\min} and the minimum mixing weight is bounded from below by α_{\min} . We note that while σ can also be estimated, we will assume that it is known in advance to simplify the arguments. The number of components needs to be known in advance which is in line with other work on the subject. Our main result is an algorithm guaranteed to produce an approximating mixture \tilde{p} , whose means and mixing coefficients are all within ϵ of their true values and whose running time is a polynomial in all parameters other than k . Input to our algorithm is $\alpha_{\min}, \sigma, k, N$ points in \mathbb{R}^n sampled from p and an arbitrary small positive ϵ satisfying $\epsilon \leq \frac{d_{\min}}{2}$. The algorithm has the following main steps.

Parameters: $\alpha_{\min}, d_{\min}, \sigma, k$.

Input: $\epsilon \leq \frac{d_{\min}}{2}, N$ points in \mathbb{R}^n sampled from p .

Output: $\boldsymbol{\theta}^*$, the vector of approximated means and mixing coefficients.

Step 1. (Reduction to k dimensions). Given a polynomial number of data points sampled from p it is possible to identify the k -dimensional span of the means $\boldsymbol{\mu}_i$ in \mathbb{R}^n by using Singular Value Decomposition (see (Vempala & Wang, 2002)). By an additional argument the problem can be reduced to analyzing a mixture of k Gaussians in \mathbb{R}^k .

Step 2. (Construction of kernel density estimator). Using Step 1, we can assume that $n = k$. Given a sample of N points in \mathbb{R}^k , we construct a density function p_{kde} using an appropriately chosen kernel density estimator. Given sufficiently many points, $\|p - p_{kde}\|$ can be made arbitrarily small. Note that while p_{kde} is a mixture of Gaussians, it is *not* a mixture of k Gaussians.

Step 3. (Grid search). Let $\Theta = (\mathbb{R}^k)^k \times \mathbb{R}^k$ be the $k^2 + k$ -dimensional space of parameters (component means and mixing coefficients) to be estimated. Because of Step 1, we can assume (see Lemma 1) $\boldsymbol{\mu}_i$ s are in \mathbb{R}^k .

For any $\tilde{\boldsymbol{\theta}} = (\tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_2, \dots, \tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\alpha}}) = (\tilde{\mathbf{m}}, \tilde{\boldsymbol{\alpha}}) \in \Theta$, let $p(\mathbf{x}, \tilde{\boldsymbol{\theta}})$ be the corresponding mixture distribution. Note that $\boldsymbol{\theta} = (\mathbf{m}, \boldsymbol{\alpha}) \in \Theta$ are the true parameters. We obtain a value G (polynomial in all arguments for a fixed k) from Theorem 4 and take a grid M_G of size G in Θ . The value $\boldsymbol{\theta}^*$ is found from a grid search according to the following equation

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\tilde{\boldsymbol{\theta}} \in M_G} \left\{ \|p(\mathbf{x}, \tilde{\boldsymbol{\theta}}) - p_{kde}\| \right\} \quad (1)$$

We show that the means and mixing coefficients obtained by taking $\boldsymbol{\theta}^*$ are close to the true underlying means and mixing coefficients of p with high probability. We note that our algorithm is deterministic and the uncertainty comes only from the sample (through the SVD projection and density estimation).

While a somewhat different grid search algorithm was used in (Feldman et al., 2006), the main novelty of our result is showing that the parameters estimated from the grid search are close to the true underlying parameters of the mixture. In principle, it is conceivable that two different configurations of Gaussians could give rise to very similar mixture distributions. However, we show that this is not the case. Specifically, and this is the theoretical core of this paper, we show that mixtures with different means/mixing coefficients cannot be close in L^2 norm³ (Theorem 2) and thus the grid search yields parameter values $\boldsymbol{\theta}^*$ that are close to the true values of the means and mixing coefficients.

To provide a better high-level overview of the whole proof we give a high level summary of the argument (Steps 2 and 3).

1. Since we do not know the underlying probability distribution p directly, we construct p_{kde} , which is a proxy for $p = p(\mathbf{x}, \boldsymbol{\theta})$. p_{kde} is obtained by taking an appropriate non-parametric density estimate and, given a sufficiently large polynomial sample, can be made to be arbitrarily close to p in L^2 norm (see Lemma 9). Thus the problem of approximating p in L^2 norm can be replaced by approximating p_{kde} .
2. The main technical part of the paper are the lower and upper bounds on the norm $\|p(\mathbf{x}, \boldsymbol{\theta}) - p(\mathbf{x}, \tilde{\boldsymbol{\theta}})\|$ in terms of the Hausdorff distance between the component means (considered as sets of k points) \mathbf{m} and $\tilde{\mathbf{m}}$. Specifically, in Theorem 2 and Lemma 3 we prove that for $\tilde{\boldsymbol{\theta}} = (\tilde{\mathbf{m}}, \tilde{\boldsymbol{\alpha}})$

$$d_H(\mathbf{m}, \tilde{\mathbf{m}}) \leq f(\|p(\mathbf{x}, \boldsymbol{\theta}) - p(\mathbf{x}, \tilde{\boldsymbol{\theta}})\|) \leq h(d_H(\mathbf{m}, \tilde{\mathbf{m}}) + \|\boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}}\|_1)$$

where f, h are some explicitly given increasing functions. The lower bound shows that $d_H(\mathbf{m}, \tilde{\mathbf{m}})$ can be controlled by making $\|p(\mathbf{x}, \boldsymbol{\theta}) - p(\mathbf{x}, \tilde{\boldsymbol{\theta}})\|$ sufficiently small, which (assuming minimum separation

³Note that our notion of distance between two density functions is slightly different from the standard ones used in literature, e.g., Hellinger distance or KL divergence. However, our goal is to estimate the parameters and here we use L^2 norm merely as a tool to describe that two distributions are different.

d_{min} between the components of p) immediately implies that each component mean of \mathbf{m} is close to exactly one component mean of $\tilde{\mathbf{m}}$.

On the other hand, the upper bound guarantees that a search over a sufficiently fine grid in the space Θ will produce a value θ^* , s.t. $\|p(\mathbf{x}, \theta) - p(\mathbf{x}, \theta^*)\|$ is small.

3. Once the component means \mathbf{m} and $\tilde{\mathbf{m}}$ are shown to be close an argument using the Lipschitz property of the mixture with respect to the mean locations can be used to establish that the corresponding mixing coefficient are also close (Corollary 5).

We will now briefly outline the argument for the main theoretical contribution of this paper which is a lower bound on the L^2 norm in terms of the Hausdorff distance (Theorem 2).

1. (Minimum distance, reduction from \mathbb{R}^k to \mathbb{R}^1) Suppose a component mean μ_i , is separated from every estimated mean $\tilde{\mu}_j$ by a distance of at least d , then there exists a unit vector \mathbf{v} in \mathbb{R}^k such that $\forall_{i,j} |\langle \mathbf{v}, (\tilde{\mu}_i - \mu_j) \rangle| \geq \frac{d}{4k^2}$. In other words a certain amount of separation is preserved after an appropriate projection to one dimension. See Lemma 10 for a proof.
2. (Norm estimation, reduction from \mathbb{R}^k to \mathbb{R}^1). Let p and \tilde{p} be the true and estimated density respectively and let \mathbf{v} be a unit vector in \mathbb{R}^k . $p_{\mathbf{v}}$ and $\tilde{p}_{\mathbf{v}}$ will denote the one-dimensional marginal densities obtained by integrating p and \tilde{p} in the directions orthogonal to \mathbf{v} . It is easy to see that $p_{\mathbf{v}}$ and $\tilde{p}_{\mathbf{v}}$ are mixtures of 1-dimensional Gaussians, whose means are projections of the original means onto \mathbf{v} . It is shown in Lemma 11 that

$$\|p - \tilde{p}\|^2 \geq \left(\frac{1}{c\sigma}\right)^k \|p_{\mathbf{v}} - \tilde{p}_{\mathbf{v}}\|^2$$

and thus to provide a lower bound for $\|p - \tilde{p}\|$ it is sufficient to provide an analogous bound (with a different separation between the means) in one dimension.

3. (1-d lower bound) Finally, we consider a mixture q of $2k$ Gaussians in one dimension, with the assumption that one of the component means is separated from the rest of the component means by at least t and that the (not necessarily positive) mixing weights exceed α_{min} in absolute value. Assuming that the means lie in an interval $[-a, a]$ we show (Theorem 6)

$$\|q\|^2 \geq \alpha_{min}^{4k} \left(\frac{t}{a^2}\right)^{Ck^2}$$

for some positive constant C independent of k .

The proof of this result relies on analyzing the Taylor series for the Fourier transform of q near zeros, which turns out to be closely related to a certain Vandermonde matrix.

Combining 1 and 2 above and applying the result in 3, $q = p_{\mathbf{v}} - \tilde{p}_{\mathbf{v}}$ yields the desired lower bound for $\|p - \tilde{p}\|$.

3 Main Results

In this section we present our main results. First we show that we can reduce the problem in \mathbb{R}^n to a corresponding problem in \mathbb{R}^k , where n represents the dimension and k is the number of components, at the cost of an arbitrarily small error. Then we solve the reduced problem in \mathbb{R}^k , again allowing for only an arbitrarily small error, by establishing appropriate lower and upper bounds of a mixture norm in \mathbb{R}^k .

Lemma 1 (Reduction from \mathbb{R}^n to \mathbb{R}^k) Consider a mixture of k n -dimensional spherical Gaussians $p(\mathbf{x}) = \sum_{i=1}^k \alpha_i K(\mathbf{x}, \mu_i)$ where the means lie within a cube $[-1, 1]^n$, $\|\mu_i - \mu_j\| \geq d_{min} > 0, \forall_{i \neq j}$ and for all i , $\alpha_i > \alpha_{min}$. For any positive $\epsilon \leq \frac{d_{min}}{2}$ and $\delta \in (0, 1)$, given a sample of size $\text{poly}\left(\frac{n}{\epsilon \alpha_{min}}\right) \cdot \log\left(\frac{1}{\delta}\right)$, with probability greater than $1 - \delta$, the problem of learning the parameters (means and mixing weights) of p within ϵ error can be reduced to learning the parameters of a k -dimensional mixture of spherical Gaussians $p_o(\mathbf{x}) = \sum_{i=1}^k \alpha_i K(\mathbf{x}, \nu_i)$ where the means lie within a cube $[-\sqrt{\frac{n}{k}}, \sqrt{\frac{n}{k}}]^k$, $\|\nu_i - \nu_j\| > \frac{d_{min}}{2} > 0, \forall_{i \neq j}$. However, in \mathbb{R}^k we need to learn the means within $\frac{\epsilon}{2}$ error.

Proof: For $i = 1, \dots, k$, let $\mathbf{v}_i \in \mathbb{R}^n$ be the top k right singular vectors of a data matrix of size $\text{poly}\left(\frac{n}{\epsilon \alpha_{min}}\right) \cdot \log\left(\frac{1}{\delta}\right)$ sampled from $p(\mathbf{x})$. It is well known (see (Vempala & Wang, 2002)) that the space spanned by the

means $\{\boldsymbol{\mu}_i\}_{i=1}^k$ remains arbitrarily close to the space spanned by $\{\boldsymbol{v}_i\}_{i=1}^k$. In particular, with probability greater than $1 - \delta$, the projected means $\{\tilde{\boldsymbol{\mu}}_i\}_{i=1}^k$ satisfy $\|\boldsymbol{\mu}_i - \tilde{\boldsymbol{\mu}}_i\| \leq \frac{\epsilon}{2}$ for all i (see Lemma 12).

Note that each projected mean $\tilde{\boldsymbol{\mu}}_i \in \mathbb{R}^n$ can be represented by a k dimensional vector $\boldsymbol{\nu}_i$ which are the coefficients along the singular vectors \boldsymbol{v}_j s, that is for all i , $\tilde{\boldsymbol{\mu}}_i = \sum_{j=1}^k \nu_{ij} \boldsymbol{v}_j$. Thus, for any $i \neq j$, $\|\tilde{\boldsymbol{\mu}}_i - \tilde{\boldsymbol{\mu}}_j\| = \|\boldsymbol{\nu}_i - \boldsymbol{\nu}_j\|$. Since $\|\tilde{\boldsymbol{\mu}}_i - \tilde{\boldsymbol{\mu}}_j\| \geq d_{\min} - \frac{\epsilon}{2} - \frac{\epsilon}{2} = d_{\min} - \epsilon \geq d_{\min} - \frac{d_{\min}}{2} = \frac{d_{\min}}{2}$, we have $\|\boldsymbol{\nu}_i - \boldsymbol{\nu}_j\| \geq \frac{d_{\min}}{2}$. Also note that each $\boldsymbol{\nu}_i$ lie within a cube of $[-\sqrt{\frac{n}{k}}, \sqrt{\frac{n}{k}}]^k$ where the axes of the cube are along the top k singular vectors \boldsymbol{v}_j s.

Now suppose we can estimate each $\boldsymbol{\nu}_i$ by $\tilde{\boldsymbol{\nu}}_i \in \mathbb{R}^k$ such that $\|\boldsymbol{\nu}_i - \tilde{\boldsymbol{\nu}}_i\| \leq \frac{\epsilon}{2}$. Again each $\tilde{\boldsymbol{\nu}}_i$ has a corresponding representation $\hat{\boldsymbol{\mu}}_i \in \mathbb{R}^n$ such that $\hat{\boldsymbol{\mu}}_i = \sum_{j=1}^k \tilde{\nu}_{ij} \boldsymbol{v}_j$ and $\|\tilde{\boldsymbol{\mu}}_i - \hat{\boldsymbol{\mu}}_i\| = \|\boldsymbol{\nu}_i - \tilde{\boldsymbol{\nu}}_i\|$. This implies for each i , $\|\boldsymbol{\mu}_i - \hat{\boldsymbol{\mu}}_i\| \leq \|\boldsymbol{\mu}_i - \tilde{\boldsymbol{\mu}}_i\| + \|\tilde{\boldsymbol{\mu}}_i - \hat{\boldsymbol{\mu}}_i\| \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$. ■

From here onwards we will deal with mixture of Gaussians in \mathbb{R}^k . Thus we will assume that p_o denotes the true mixture with means $\{\boldsymbol{\nu}_i\}_{i=1}^k$ while \tilde{p}_o represents any other mixture in \mathbb{R}^k with different means and mixing weights.

We first prove a lower bound for $\|p_o - \tilde{p}_o\|$.

Theorem 2 (Lower bound in \mathbb{R}^k) Consider a mixture of k k -dimensional spherical Gaussians $p_o(\boldsymbol{x}) = \sum_{i=1}^k \alpha_i K(\boldsymbol{x}, \boldsymbol{\nu}_i)$ where the means lie within a cube $[-\sqrt{\frac{n}{k}}, \sqrt{\frac{n}{k}}]^k$, $\|\boldsymbol{\nu}_i - \boldsymbol{\nu}_j\| \geq \frac{d_{\min}}{2} > 0, \forall i \neq j$ and for all $i, \alpha_i > \alpha_{\min}$. Let $\tilde{p}_o(\boldsymbol{x}) = \sum_{i=1}^k \tilde{\alpha}_i K(\boldsymbol{x}, \tilde{\boldsymbol{\nu}}_i)$ be some arbitrary mixture such that the Hausdorff distance between the set of true means \boldsymbol{m} and the estimated means $\tilde{\boldsymbol{m}}$ satisfies $d_H(\boldsymbol{m}, \tilde{\boldsymbol{m}}) \leq \frac{d_{\min}}{4}$. Then $\|p_o - \tilde{p}_o\|^2 \geq \left(\frac{\alpha_{\min}^4}{c\sigma}\right)^k \left(\frac{d_H(\boldsymbol{m}, \tilde{\boldsymbol{m}})}{4nk^2}\right)^{Ck^2}$ where C, c are some positive constants independent of n, k .

Proof: Consider any arbitrary $\boldsymbol{\nu}_i$ such that its closest estimate $\tilde{\boldsymbol{\nu}}_i$ from $\tilde{\boldsymbol{m}}$ is $t = \|\boldsymbol{\nu}_i - \tilde{\boldsymbol{\nu}}_i\|$. Note that $t \leq \frac{d_{\min}}{4}$ and all other $\boldsymbol{\nu}_j, \tilde{\boldsymbol{\nu}}_j, j \neq i$ are at a distance at least t from $\boldsymbol{\nu}_i$. Lemma 10 ensures the existence of a direction $\boldsymbol{v} \in \mathbb{R}^k$ such that upon projecting on which $|\langle \boldsymbol{v}, (\boldsymbol{\nu}_i - \tilde{\boldsymbol{\nu}}_i) \rangle| \geq \frac{t}{4k^2}$ and all other projected means $\langle \boldsymbol{v}, \boldsymbol{\nu}_j \rangle, \langle \boldsymbol{v}, \tilde{\boldsymbol{\nu}}_j \rangle, j \neq i$ are at a distance at least $\frac{t}{4k^2}$ from $\langle \boldsymbol{v}, \boldsymbol{\nu}_i \rangle$. Note that after projecting on \boldsymbol{v} , the mixture becomes a mixture of 1-dimensional Gaussians with variance σ^2 and whose projected means lie within $[-\sqrt{n}, \sqrt{n}]$. Let us denote these 1-dimensional mixtures by p_v and \tilde{p}_v respectively. Then using Theorem 6 $\|p_v - \tilde{p}_v\|^2 \geq \alpha_{\min}^{4k} \left(\frac{(t/4k^2)}{n}\right)^{Ck^2}$. Note that we obtain p_v (respectively \tilde{p}_v) by integrating p_o (respectively \tilde{p}_o) in all $(k-1)$ orthogonal directions to \boldsymbol{v} . Now we need to relate $\|p_o - \tilde{p}_o\|$ and $\|p_v - \tilde{p}_v\|$. This is done in Lemma 11 to ensure that $\|p_o - \tilde{p}_o\|^2 \geq \left(\frac{1}{c\sigma}\right)^k \|p_v - \tilde{p}_v\|^2$ where $c >$ is in chosen such a way that in any arbitrary direction probability mass of each projected Gaussian on that direction becomes negligible outside the interval of $[-c\sigma/2, c\sigma/2]$. Thus, $\|p_o - \tilde{p}_o\|^2 \geq \left(\frac{\alpha_{\min}^4}{c\sigma}\right)^k \left(\frac{t}{4nk^2}\right)^{Ck^2}$. Since this holds for any arbitrary $\boldsymbol{\nu}_i$, we can replace t by $d_H(\boldsymbol{m}, \tilde{\boldsymbol{m}})$. ■

Next, we prove a straightforward upper bound for $\|p_o - \tilde{p}_o\|$.

Lemma 3 (Upper bound in \mathbb{R}^k) Consider a mixture of k , k -dimensional spherical Gaussians $p_o(\boldsymbol{x}) = \sum_{i=1}^k \alpha_i K(\boldsymbol{x}, \boldsymbol{\nu}_i)$ where the means lie within a cube $[-\sqrt{\frac{n}{k}}, \sqrt{\frac{n}{k}}]^k$, $\|\boldsymbol{\nu}_i - \boldsymbol{\nu}_j\| \geq \frac{d_{\min}}{2} > 0, \forall i \neq j$ and for all $i, \alpha_i > \alpha_{\min}$. Let $\tilde{p}_o(\boldsymbol{x}) = \sum_{i=1}^k \tilde{\alpha}_i K(\boldsymbol{x}, \tilde{\boldsymbol{\nu}}_i)$ be some arbitrary mixture such that the Hausdorff distance between the set of true means \boldsymbol{m} and the estimated means $\tilde{\boldsymbol{m}}$ satisfies $d_H(\boldsymbol{m}, \tilde{\boldsymbol{m}}) \leq \frac{d_{\min}}{4}$. Then there exists a permutation $\pi : \{1, 2, \dots, k\} \rightarrow \{1, 2, \dots, k\}$ such that

$$\|p_o - \tilde{p}_o\| \leq \frac{1}{(2\pi\sigma^2)^{k/2}} \sum_{i=1}^k \left(\sqrt{|\alpha_i - \tilde{\alpha}_{\pi(i)}|^2 + \frac{d_H^2(\boldsymbol{m}, \tilde{\boldsymbol{m}})}{\sigma^2}} \right)$$

Proof: Due to the constraint on the Hausdorff distance and constraint on the pair wise distance between the means of \boldsymbol{m} , there exists a permutation $\pi : \{1, 2, \dots, k\} \rightarrow \{1, 2, \dots, k\}$ such that $\|\boldsymbol{\nu}_i - \tilde{\boldsymbol{\nu}}_{\pi(i)}\| \leq d_H(\boldsymbol{m}, \tilde{\boldsymbol{m}})$. Due to one-to-one correspondence, without loss of generality we can write,

$$\begin{aligned} \|p_o - \tilde{p}_o\| &\leq \sum_{i=1}^k \|g_i\| \text{ where } g_i(\boldsymbol{x}) = \alpha_i K(\boldsymbol{x}, \boldsymbol{\nu}_i) - \tilde{\alpha}_{\pi(i)} K(\boldsymbol{x}, \tilde{\boldsymbol{\nu}}_{\pi(i)}). \text{ Now using Lemma 13,} \\ \|g_i\|^2 &\leq \frac{1}{(2\pi\sigma^2)^k} \left(\alpha_i^2 + \tilde{\alpha}_{\pi(i)}^2 - 2\alpha_i \tilde{\alpha}_{\pi(i)} \exp\left(-\frac{\|\boldsymbol{\nu}_i - \tilde{\boldsymbol{\nu}}_{\pi(i)}\|^2}{2\sigma^2}\right) \right) \\ &= \frac{1}{(2\pi\sigma^2)^k} \left((\alpha_i - \tilde{\alpha}_{\pi(i)})^2 + 2\alpha_i \tilde{\alpha}_{\pi(i)} \left(1 - \exp\left(-\frac{\|\boldsymbol{\nu}_i - \tilde{\boldsymbol{\nu}}_{\pi(i)}\|^2}{2\sigma^2}\right)\right) \right) \\ &\leq \frac{1}{(2\pi\sigma^2)^k} \left((\alpha_i - \tilde{\alpha}_{\pi(i)})^2 + \frac{\alpha_i \tilde{\alpha}_{\pi(i)} \|\boldsymbol{\nu}_i - \tilde{\boldsymbol{\nu}}_{\pi(i)}\|^2}{\sigma^2} \right) \end{aligned}$$

We now present our main result for learning mixture of Gaussians with arbitrary small separation. ■

Theorem 4 Consider a mixture of k n -dimensional spherical Gaussians $p(\mathbf{x}) = \sum_{i=1}^k \alpha_i K(\mathbf{x}, \boldsymbol{\mu}_i)$ where the means lie within a cube $[-1, 1]^n$, $\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\| > d_{\min} > 0, \forall i \neq j$ and for all i , $\alpha_i > \alpha_{\min}$. Then given any positive $\epsilon \leq \frac{d_{\min}}{2}$ and $\delta \in (0, 1)$, there exists a positive C_1 independent of n and k such that using a sample of size $N = \text{poly} \left(\left(\frac{nk^2}{\epsilon} \right)^{k^3} \cdot \log^k \left(\frac{2}{\delta} \right) \right)$ and a grid M_G of size $G = \frac{(\alpha_{\min}^4)^k}{k^{3/2}} \left(\frac{\epsilon}{8nk^2} \right)^{C_1 k^2}$, our algorithm given by Equation 1 runs in time $\frac{k^{3/2}}{(\alpha_{\min}^4 \sigma)^k} \left(\frac{n^{3/2} k^{1/2}}{\epsilon} \right)^{C_1 k^2}$ and provides mean estimates which, with probability greater than $1 - \delta$, are within ϵ of their corresponding true values.

Proof: The proof has several parts.

SVD projection: We have shown in Lemma 1 that after projecting to SVD space (using a sample of size $\text{poly} \left(\frac{n}{\alpha_{\min} \epsilon} \right) \cdot \log \left(\frac{2}{\delta} \right)$), we need to estimate the parameters of the mixture in \mathbb{R}^k , $p_o(\mathbf{x}) = \sum_{i=1}^k \alpha_i K(\mathbf{x}, \boldsymbol{\nu}_i)$ where we must estimate the means within $\frac{\epsilon}{2}$ error.

Grid Search: Let us denote the parameters⁴ of the underlying mixture $p_o(\mathbf{x}, \boldsymbol{\theta})$ by $\boldsymbol{\theta} = (\mathbf{m}, \boldsymbol{\alpha}) = (\boldsymbol{\nu}_1, \dots, \boldsymbol{\nu}_k, \boldsymbol{\alpha}) \in \mathbb{R}^{k^2+k}$ and any approximating mixture $p_o(\mathbf{x}, \tilde{\boldsymbol{\theta}})$ has parameters $\tilde{\boldsymbol{\theta}} = (\tilde{\mathbf{m}}, \tilde{\boldsymbol{\alpha}})$. We have proved the bounds $f_1(d_H(\mathbf{m}, \tilde{\mathbf{m}})) \leq \|p(\mathbf{x}, \boldsymbol{\theta}) - p(\mathbf{x}, \tilde{\boldsymbol{\theta}})\| \leq f_2(d_H(\mathbf{m}, \tilde{\mathbf{m}}) + \|\boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}}\|_1)$ (see Theorem 2, Lemma 3), where f_1 and f_2 are increasing functions. Let G be the step/grid size (whose value we need to set) that we use for gridding along each of the $k^2 + k$ parameters over the grid M_G . We note that the L^2 norm of the difference can be computed efficiently by multidimensional trapezoidal rule or any other standard numerical analysis technique (see e.g., (Burden & Faires, 1993)). Since this integration needs to be performed on a $(k^2 + k)$ -dimensional space, for any pre-specified precision parameter ϵ , this can be done in time $\left(\frac{1}{\epsilon}\right)^{O(k^2)}$. Now note that there exists a point $\boldsymbol{\theta}^* = (\mathbf{m}^*, \boldsymbol{\alpha}^*)$ on the grid M_G , such that if somehow we can identify this point as our parameter estimate then we make an error at most $G/2$ in estimating each mixing weight and make an error at most $G\sqrt{k}/2$ in estimating each mean. Since there are k mixing weights and k means to be estimated, $\|p_o(\mathbf{x}, \boldsymbol{\theta}) - p_o(\mathbf{x}, \boldsymbol{\theta}^*)\| \leq f_2(d_H(\mathbf{m}, \mathbf{m}^*) + \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|_1) \leq f_2(G) = \frac{k\sqrt{1+k/\sigma^2}}{2(2\pi\sigma^2)^{k/2}} G$. Thus,

$$f_1(d_H(\mathbf{m}, \mathbf{m}^*)) \leq \|p_o(\mathbf{x}, \boldsymbol{\theta}) - p_o(\mathbf{x}, \boldsymbol{\theta}^*)\| \leq f_2(G)$$

Now, according to Lemma 9, using a sample of size $\Omega \left(\left[\frac{\log(2/\delta)}{\epsilon_*^2} \right]^k \right)$ we can obtain a kernel density estimate such that with probability greater than $1 - \frac{\delta}{2}$,

$$\|p_{kde} - p_o(\mathbf{x}, \boldsymbol{\theta})\| \leq \epsilon_* \quad (2)$$

By triangular inequality this implies,

$$f_1(d_H(\mathbf{m}, \mathbf{m}^*)) - \epsilon_* \leq \|p_{kde} - p_o(\mathbf{x}, \boldsymbol{\theta}^*)\| \leq f_2(G) + \epsilon_* \quad (3)$$

Since there is a one-to-one correspondence between the set of means of \mathbf{m} and \mathbf{m}^* , $d_H(\mathbf{m}, \mathbf{m}^*)$ essentially provides the maximum estimation error for any pair of true mean and its corresponding estimate. Suppose we choose G such that it satisfies

$$2\epsilon_* + f_2(G) \leq f_1\left(\frac{\epsilon}{2}\right) \quad (4)$$

For this choice of grid size, Equation 3 and Equation 4 ensures that $f_1(d_H(\mathbf{m}, \mathbf{m}^*)) \leq f_2(G) + 2\epsilon_* \leq f_1\left(\frac{\epsilon}{2}\right)$. Hence $d_H(\mathbf{m}, \mathbf{m}^*) \leq \frac{\epsilon}{2}$. Now consider a point $\boldsymbol{\theta}^N = (\mathbf{m}^N, \boldsymbol{\alpha}^N)$ on the grid M_G such that $d_H(\mathbf{m}, \mathbf{m}^N) > \frac{\epsilon}{2}$. This implies,

$$f_1(d_H(\mathbf{m}, \mathbf{m}^N)) > f_1\left(\frac{\epsilon}{2}\right) \quad (5)$$

Now,

$$\begin{aligned} \|p_o(\mathbf{x}, \boldsymbol{\theta}^N) - p_{kde}\| &\stackrel{a}{\geq} \|p_o(\mathbf{x}, \boldsymbol{\theta}^N) - p_o(\mathbf{x}, \boldsymbol{\theta})\| - \|p_o(\mathbf{x}, \boldsymbol{\theta}) - p_{kde}\| \\ &\stackrel{b}{\geq} f_1(d_H(\mathbf{m}, \mathbf{m}^N)) - \epsilon_* \\ &\stackrel{c}{>} f_1\left(\frac{\epsilon}{2}\right) - \epsilon_* \\ &\stackrel{d}{\geq} f_2(G) + \epsilon_* \\ &\stackrel{e}{\geq} \|p_o(\mathbf{x}, \boldsymbol{\theta}^*) - p_{kde}\| \end{aligned}$$

⁴To make our presentation simple we assume that the single parameter variance is fixed and known. Note that it can also be estimated.

where, inequality a follows from triangular inequality, inequality b follows from Equation 2, strict inequality c follows from Equation 5, inequality d follows from Equation 4 and finally inequality e follows from Equation 3. Setting $\epsilon_* = \frac{1}{3}f_1\left(\frac{\epsilon}{2}\right)$, Equation 4 and the above strict inequality guarantees that for a choice of Grid size $G = f_2^{-1}\left(\frac{1}{3}f_1\left(\frac{\epsilon}{2}\right)\right) = \left(\frac{\alpha_{\min}^{4k}}{k^{3/2}}\right)\left(\frac{\epsilon}{8nk^2}\right)^{C_1k^2}$ the solution obtained by equation 1 can have mean estimation error at most $\frac{\epsilon}{2}$. Once projected onto SVD space each projected mean lies within a cube $[-\sqrt{\frac{n}{k}}, \sqrt{\frac{n}{k}}]^k$. With the above chosen grid size, grid search for the means runs in time $\left(\frac{k^{3/2}}{\alpha_{\min}^{4k}}\right) \cdot \left(\frac{n^{3/2}k^{1/2}}{\epsilon}\right)^{C_1k^2}$. Note that grid search for the mixing weights runs in time $\left(\frac{k^{3/2}}{\alpha_{\min}^{4k}}\right) \cdot \left(\frac{nk^2}{\epsilon}\right)^{C_1k^2}$. ■

We now show that not only the mean estimates but also the mixing weights obtained by solving Equation 1 satisfy $|\alpha_i - \tilde{\alpha}_i| \leq \epsilon$ for all i . In particular we show that if two mixtures have almost same means and the L^2 norm of difference of their densities is small then the difference of the corresponding mixing weights must also be small.

Corollary 5 *With sample size and grid size as in Theorem 4, the solution of Equation 1 provides mixing weight estimates which are, with high probability, within ϵ of their true values.*

Due to space limitation we defer the proof is omitted.

3.1 Lower Bound in 1-Dimensional Setting

In this section we provide the proof of our main theoretical result in 1-dimensional setting. Before we present the actual proof, we provide high level arguments that lead us to this result. First note that Fourier transform of a mixture of k univariate Gaussians $q(x) = \sum_{i=1}^k \alpha_i K(x, \mu_i)$ is given by

$$\begin{aligned} \mathcal{F}(q)(u) &= \frac{1}{\sqrt{2\pi}} \int q(x) \exp(-iux) dx = \frac{1}{\sqrt{2\pi}} \sum_{j=1}^k \alpha_j \exp\left(-\frac{1}{2}(\sigma^2 u^2 + i2u\mu_j)\right) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\sigma^2 u^2}{2}\right) \sum_{j=1}^k \alpha_j \exp(-iu\mu_j) \end{aligned}$$

Thus, $\|\mathcal{F}(q)\|^2 = \frac{1}{2\pi} \int |\sum_{j=1}^k \alpha_j \exp(-iu\mu_j)|^2 \exp(-\sigma^2 u^2) du$. Since L^2 norm of a function and its Fourier transform are the same, we can write,

$$\|q\|^2 = \frac{1}{2\pi} \int |\sum_{j=1}^k \alpha_j \exp(-iu\mu_j)|^2 \exp(-\sigma^2 u^2) du.$$

Further, $\frac{1}{2\pi} \int |\sum_{j=1}^k \alpha_j \exp(-iu\mu_j)|^2 \exp(-\sigma^2 u^2) du = \frac{1}{2\pi} \int |\sum_{j=1}^k \alpha_j \exp(iu\mu_j)|^2 \exp(-\sigma^2 u^2) du$ and we can write,

$$\|q\|^2 = \frac{1}{2\pi} \int |g(u)|^2 \exp(-\sigma^2 u^2) du$$

where $g(u) = \sum_{j=1}^k \alpha_j \exp(i\mu_j u)$. This a complex valued function of a real variable which is infinitely differentiable everywhere. In order to bound the above square norm from below, now our goal is to find an interval where $|g(u)|^2$ is bounded away from zero. In order to achieve this, we write Taylor series expansion of $g(u)$ at the origin using $(k-1)$ terms. This can be written in matrix vector multiplication format $g(u) = \mathbf{u}^t \mathbf{A} \boldsymbol{\alpha} + O(u^k)$, where $\mathbf{u}^t = [1 \ u \ \frac{u^2}{2!} \ \cdots \ \frac{u^{k-1}}{(k-1)!}]$, such that $\mathbf{A} \boldsymbol{\alpha}$ captures the function value and $(k-1)$ derivative values at origin. In particular, $\|\mathbf{A} \boldsymbol{\alpha}\|^2$ is the sum of the squares of the function g and $k-1$ derivatives at origin. Noting that \mathbf{A} is a Vandermonde matrix we establish (see Lemma 16) $\|\mathbf{A} \boldsymbol{\alpha}\| \geq \alpha_{\min} \left(\frac{t}{2\sqrt{n}}\right)^{k-1}$. This implies that at least one of the $(k-1)$ derivatives, say the j^{th} one, of g is bounded away from zero at origin. Once this fact is established, and noting that $(j+1)^{th}$ derivative of g is bounded from above everywhere, it is easy to show (see Lemma 14) that it is possible to find an interval $(0, a)$ where j^{th} derivative of g is bounded away from zero in this whole interval. Then using Lemma 15, it can be shown that, it is possible to find a subinterval of $(0, a)$ where the $(j-1)^{th}$ derivative of g is bounded away from zero. And thus, successively repeating this Lemma j times, it is easy to show that there exists a subinterval of $(0, a)$ where $|g|$ is bounded away from zero. Once this subinterval is found, it is easy to show that $\|q\|^2$ is lower bounded as well.

Now we present the formal statement of our result.

Theorem 6 (Lower bound in \mathbb{R}) *Consider a mixture of k univariate Gaussians $q(x) = \sum_{i=1}^k \alpha_i K(x, \mu_i)$ where, for all i , the mixing coefficients $\alpha_i \in (-1, 1)$ and the means $\mu_i \in [-\sqrt{n}, \sqrt{n}]$. Suppose there exists a μ_l such that $\min_j |\mu_l - \mu_j| \geq t$, and for all i , $|\alpha_i| \geq \alpha_{\min}$. Then the L^2 norm of q satisfies $\|q\|^2 \geq \alpha_{\min}^{2k} \left(\frac{t}{n}\right)^{Ck^2}$ where C is some positive constant independent of k .*

Proof: Note that,

$$\|g\|^2 = \frac{1}{2\pi} \int |g(u)|^2 \exp(-\sigma^2 u^2) du$$

where, $g(u) = \sum_{j=1}^k \alpha_j \exp(i\mu_j u)$. Thus, in order to bound the above square norm from below, we need to find an interval where $g(u)$ is bounded away from zero. Note that $g(u)$ is an infinitely differentiable function with n^{th} order derivative $g^{(n)}(u) = \sum_{j=1}^k \alpha_j (i\mu_j)^n \exp(i\mu_j u)$. Now we can write the Taylor series expansion of $g(u)$ about origin as,

$$g(u) = g(0) + g^{(1)}(0) \frac{u}{1!} + g^{(2)}(0) \frac{u^2}{2!} + \dots + g^{(k-1)}(0) \frac{u^{(k-1)}}{(k-1)!} + O(u^k)$$

which can be written as

$$g(u) = \begin{bmatrix} 1 & u & \frac{u^2}{2!} & \dots & \frac{u^{k-1}}{(k-1)!} \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ i\mu_1 & i\mu_2 & i\mu_3 & \dots & i\mu_k \\ (i\mu_1)^2 & (i\mu_2)^2 & (i\mu_3)^2 & \dots & (i\mu_k)^2 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ (i\mu_1)^{k-1} & (i\mu_2)^{k-1} & (i\mu_3)^{k-1} & \dots & (i\mu_k)^{k-1} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_k \end{bmatrix}}_{\boldsymbol{\alpha}} + O(u^k)$$

Note that matrix \mathbf{A} is Vandermonde matrix thus, using Lemma 16 this implies $|g(0)|^2 + |g^{(1)}(0)|^2 + \dots + |g^{(k-1)}(0)|^2 = \|\mathbf{A}\boldsymbol{\alpha}\|^2 \geq \alpha_{\min}^2 \left(\frac{t}{1+\sqrt{n}}\right)^{2(k-1)} \geq \alpha_{\min}^2 \left(\frac{t}{2\sqrt{n}}\right)^{2(k-1)}$. This further implies that either $|g(0)|^2 \geq \frac{\alpha_{\min}^2}{k} \left(\frac{t}{2\sqrt{n}}\right)^{2(k-1)}$ or there exists a $j \in \{1, 2, \dots, k-1\}$ such that $|g^{(j)}(0)|^2 \geq \frac{\alpha_{\min}^2}{k} \left(\frac{t}{2\sqrt{n}}\right)^{2(k-1)}$. In the worst case we can have $j = k-1$, i.e. the $(k-1)$ -th derivative of g is lower bounded at origin and we need to find an interval where g itself is lower bounded.

Next, note that for any u , $g^{(k)}(u) = \sum_{j=1}^k \alpha_j (i\mu_j)^k \exp(iu\mu_j)$. Thus, $|g^{(k)}| \leq \sum_{j=1}^k |\alpha_j| (i\mu_j)^k \leq \alpha_{\max} (\sqrt{n})^k$. Assuming $t \leq 2\sqrt{n}$, if we let $M = \frac{\alpha_{\min}}{\sqrt{k}} \left(\frac{t}{2\sqrt{n}}\right)^k$, then using Lemma 14, if we choose $a = \frac{M}{2\sqrt{2}\alpha_{\max}(\sqrt{n})^k} = \frac{\alpha_{\min}}{\alpha_{\max} 2\sqrt{2}k} \left(\frac{t}{2n}\right)^k$, and thus, in the interval $[0, a]$, $|g^{(k-1)}| > \frac{M}{2} = \frac{\alpha_{\min}}{2\sqrt{k}} \left(\frac{t}{2\sqrt{n}}\right)^k$. This implies $|Re[g^{(k-1)}]|^2 + |Im[g^{(k-1)}]|^2 > \frac{\alpha_{\min}^2}{4k} \left(\frac{t}{2\sqrt{n}}\right)^{2k}$. For simplicity denote by $h = Re[g]$, thus, $h^{(k-1)} = Re[g^{(k-1)}]$ and without loss of generality assume $|h^{(k-1)}| > \frac{\alpha_{\min}}{2\sqrt{2}k} \left(\frac{t}{2\sqrt{n}}\right)^k = \frac{M}{2\sqrt{2}}$ in the interval $(0, a)$. Now repeatedly applying Lemma 15 $(k-1)$ times yields that in the interval $\left(\frac{(3^{k-1}-1)}{3^{k-1}}a, a\right)$, (or in any other subinterval of length $\frac{a}{3^{k-1}}$ within $[0, a]$)

$$|h| > \frac{M}{2\sqrt{2}} \left(\frac{a}{6}\right) \left(\frac{a}{6.3}\right) \left(\frac{a}{6.3^2}\right) \dots \left(\frac{a}{6.3^{k-1}}\right) = \left(\frac{M}{2\sqrt{2}}\right) \left(\frac{a}{6}\right)^k \left(\frac{1}{3^{\frac{k(k-1)}{2}}}\right) = \frac{\alpha_{\max}(\sqrt{n})^k a^{k+1}}{2^k 3^{\frac{k^2+k}{2}}}$$

In particular, this implies, $|g|^2 \geq |h|^2 > \frac{\alpha_{\max}^2 n^k a^{2(k+1)}}{2^{2k} 3^{k^2+k}}$ in an interval $\left(\frac{(3^{k-1}-1)}{3^{k-1}}a, a\right)$.

Next, note that $0 < a \leq 1 \Rightarrow \exp(-\sigma^2) \leq \exp(-\sigma^2 a^2)$. Now, denoting $\beta_1 = \frac{(3^{k-1}-1)}{3^{k-1}}a$, $\beta_2 = a$, we have,

$$\begin{aligned} \|g\|^2 &\geq \frac{1}{2\pi} \int_{\beta_1}^{\beta_2} |g(u)|^2 \exp(-\sigma^2 u^2) du \geq \frac{\beta_2 - \beta_1}{2\pi} |g(\beta_2)|^2 \exp(-\sigma^2) \\ &= \left(\frac{\exp(-\sigma^2)}{2\pi}\right) \frac{\alpha_{\max}^2 n^k a^{2k+3}}{2^{2k} 3^{k^2+2k-1}} = \left(\frac{\exp(-\sigma^2) \alpha_{\min}^{2k+3}}{2\pi}\right) \left(\frac{t^{2k^2+3k}}{2^{2k^2+5k+9/2} 3^{k^2+2k-1} (\alpha_{\max})^{2k+1} k^{k+3/2} n^{2k^2+2k}}\right) \\ &\geq \left(\frac{\exp(-\sigma^2) \alpha_{\min}^{2k+3}}{2\pi}\right) \left(\frac{t^{2k^2+3k}}{2^{2k^2+5k+9/2} 3^{k^2+2k-1} k^{k+3/2} n^{2k^2+2k}}\right) \\ &\geq \alpha_{\min}^{2k} \left(\frac{t^{2k^2+3k}}{2^{O(k^2 \log n)}}\right) = \alpha_{\min}^{2k} \left(\frac{t}{n}\right)^{O(k^2)} \end{aligned}$$

where the last inequality follows from the fact that if we let,

$F(k) = 2^{2k^2+5k+9/2} 3^{k^2+2k-1} k^{k+3/2} n^{k^2+2k}$ then taking log with base 2 on both sides yields, $\log(F(k)) = (2k^2 + 5k + 9/2) + (k^2 + 2k - 1) \log 3 + (k + 3/2) \log k + (2k^2 + 2k) \log n = O(k^2 \log n)$. Thus, $F(k) = 2^{O(k^2 \log n)} = n^{O(k^2)}$. \blacksquare

⁵Note that Fourier transform is closely related to the characteristics function and the n^{th} derivative of g at origin is related to the n^{th} order moment of the mixture in the Fourier domain.

3.2 Determinant of Vandermonde Like Matrices

In this section we derive a result for the determinant of a Vandermonde-like matrix. This result will be useful in finding the angle made by any column of a Vandermonde matrix to the space spanned by the rest of the columns and will be useful in deriving the lower bound in Theorem 6.

Consider any $(n + 1) \times n$ matrix B of the form

$$B = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_n^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_1^n & x_2^n & x_3^n & \cdots & x_n^n \end{bmatrix}$$

If the last row is removed then it exactly becomes an $n \times n$ Vandermonde matrix having determinant $\prod_{i>j} (x_i - x_j)$. The interesting fact is that if any other row except the last one is removed then the corresponding $n \times n$ matrix has a structure very similar to that of a Vandermonde matrix. The following result shows how the determinants of such matrices are related to $\prod_{i>j} (x_i - x_j)$.

Lemma 7 For $1 \leq i \leq (n - 1)$, let B_i represents the $n \times n$ matrix obtained by removing the i^{th} row from B . Then $\det(B_i) = c_i \prod_{s>t} (x_s - x_t)$ where c_i is a polynomial having $\binom{n}{i-1}$ terms with each term having degree $(n - i + 1)$. Terms of the polynomial c_i represent the possible ways in which $(n - i + 1)$ x_j s can be chosen from $\{x_i\}_{i=1}^n$.

Proof: First note that if a matrix has elements that are monomials in some set of variables, then its determinant will in general be polynomial in those variables. Next, by the basic property of a determinant, that it is zero if two of its columns are same, we can deduce that for $1 \leq i < n$, $\det(B_i) = 0$ if $x_s = x_t$ for some $s \neq t$, $1 \leq s, t < n$, and hence $q_i(x_1, x_2, \dots, x_n) = \det(B_i)$ contains a factor $p(x_1, x_2, \dots, x_n) = \prod_{s>t} (x_s - x_t)$. Let $q_i(x_1, x_2, \dots, x_n) = p(x_1, x_2, \dots, x_n) r_i(x_1, x_2, \dots, x_n)$.

Now, note that each term of $p(x_1, x_2, \dots, x_n)$ has degree $0 + 1 + 2 + \dots + (n - 1) = \frac{n(n-1)}{2}$. Similarly, each term of the polynomial $q_i(x_1, x_2, \dots, x_n)$ has degree $(0 + 1 + 2 + \dots + n) - (i - 1) = \frac{n(n+1)}{2} - (i - 1)$. Hence each term of the polynomial $r_i(x_1, x_2, \dots, x_n)$ must be of degree $\frac{n(n+1)}{2} - (i - 1) - \frac{n(n-1)}{2} = (n - i + 1)$. However in each term of $r_i(x_1, x_2, \dots, x_n)$, the maximum power of any x_j can not be greater than 1. This follows from the fact that maximum power of x_j in any term of $q_i(x_1, x_2, \dots, x_n)$ is n and in any term of $p(x_1, x_2, \dots, x_n)$ is $(n - 1)$. Hence each term of $r_i(x_1, x_2, \dots, x_n)$ consists of $(n - i + 1)$ different x_j s and represents the different ways in which $(n - i + 1)$ x_j s can be chosen from $\{x_i\}_{i=1}^n$. And since it can be done in $\binom{n}{n-i+1} = \binom{n}{i-1}$ ways there will be $\binom{n}{i-1}$ terms in $r_i(x_1, x_2, \dots, x_n)$. ■

3.3 Estimation of Unknown Variance

In this section we discuss a procedure for consistent estimation of the unknown variance due to (Lindsay, 1989) (for the one-dimensional case) and prove that the estimate is polynomial. This estimated variance can then be used in place of true variance in our main algorithm discussed earlier and the remaining mixture parameters can be estimated subsequently.

We start by noting a mixture of k identical spherical Gaussians $\sum_{i=1}^k \alpha_i \mathcal{N}(\mu_i, \sigma^2 I)$ in \mathbb{R}^n projected on an arbitrary line becomes a mixture of identical 1-dimensional Gaussians $p(x) = \sum_{i=1}^k \alpha_i \mathcal{N}(\mu_i, \sigma^2)$. While the means of components may no longer be different, the variance does not change. Thus, the problem is easily reduced to the 1-dimensional case.

We will now show that the variance of a mixture of k Gaussians in 1 dimension can be estimated from a sample of size $\text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta})$, where $\epsilon > 0$ is the precision, with probability $1 - \delta$ in time $\text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta})$. This will lead to an estimate for the n -dimensional mixture using $\text{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ sample points/operations.

Consider now the set of Hermite polynomials $\gamma_i(x, \tau)$ given by the recurrence relation $\gamma_i(x, \tau) = x\gamma_{i-1}(x, \tau) - (i - 1)\tau^2\gamma_{i-2}(x, \tau)$, where $\gamma_0(x, \tau) = 1$ and $\gamma_1(x, \tau) = x$. Take M to be the $(k + 1) \times (k + 1)$ matrix defined by

$$M_{ij} = \mathbb{E}_p[\gamma_{i+j}(X, \tau)], \quad 0 \leq i + j \leq 2k.$$

It is shown in Lemma 5A of (Lindsay, 1989) that the determinant $\det(M)$ is a polynomial in τ and, moreover, that the smallest positive root of $\det(M)$, viewed as a function of τ , is equal to the variance σ of the original mixture p . We will use $d(\tau)$ to represent $\det(M)$.

This result leads to an estimation procedure, after observing that $\mathbb{E}_p[\gamma_{i+j}(X, \tau)]$ can be replaced by its empirical value given a sample X_1, X_2, \dots, X_N from the mixture distribution p . Indeed, one can construct

the empirical version of the matrix M by putting

$$\hat{M}_{ij} = \frac{1}{N} \sum_{t=1}^N [\gamma_{i+j}(X_t, \tau)], \quad 0 \leq i + j \leq 2k. \quad (6)$$

It is clear that $\hat{d}(\tau) = \det(\hat{M})(\tau)$ is a polynomial in τ . Thus we can provide an estimate σ^* for the variance σ by taking the smallest positive root of $\hat{d}(\tau)$. This leads to the following estimation procedure :

Parameter: Number of components k .

Input: N points in \mathbb{R}^n sampled from $\sum_{i=1}^k \alpha_i \mathcal{N}(\mu_i, \sigma^2 I)$.

Output: σ^* , estimate of the unknown variance.

Step 1. Select an arbitrary direction $\mathbf{v} \in \mathbb{R}^n$ and project the data points onto this direction.

Step 2. Construct the $(k+1) \times (k+1)$ matrix $\hat{M}(\tau)$ using Eq. 6

Step 3. Compute the polynomial $\hat{d}(\tau) = \det(\hat{M})(\tau)$. Obtain the estimated variance σ^* by approximating the smallest positive root of $\hat{d}(\tau)$. This can be done efficiently by using any standard numerical method or even a grid search.

We will now state our main result in this section, which establishes that this algorithm for variance estimation is indeed polynomial in both the ambient dimension n and the inverse of the desired accuracy ϵ .

Theorem 8 *For any $\epsilon > 0, 0 < \delta < 1$, if sample size $N > O\left(\frac{n^{\text{poly}(k)}}{\epsilon^2 \delta}\right)$, then the above procedure provides an estimate σ^* of the unknown variance σ such that $|\sigma - \sigma^*| \leq \epsilon$ with probability greater than $1 - \delta$.*

The idea of the proof is to show that the coefficients of the polynomials $d(\tau)$ and $\hat{d}(\tau)$ are polynomially close, given enough samples from p . That (under some additional technical conditions) can be shown to imply that the smallest positive roots of these polynomials are also close. To verify that $d(\tau)$ and $\hat{d}(\tau)$ are close, we use the fact that the coefficients of $d(\tau)$ are polynomial functions of the first $2k$ moments of p , while coefficients of $\hat{d}(\tau)$ are the same functions of the empirical moment estimates. Using standard concentration inequalities for the first $2k$ moments and providing a bound for these functions the result.

Proof: It is shown in Lemma 5A of (Lindsay, 1989) that the smallest positive root of the determinant $d(\tau) = \det(M)(\tau)$, viewed as a function of τ , is equal to the variance σ of the original mixture p and also that $d(\tau)$ undergoes a sign change at its smallest positive root. Let the smallest positive root of $\hat{d}(\tau) = \det(\hat{M})(\tau)$ be $\hat{\sigma}$. We now show for any $\epsilon > 0$ that σ and $\hat{\sigma}$ are within ϵ given $O\left(\frac{n^{\text{poly}(k)}}{\epsilon^2 \delta}\right)$ samples.

In Corollary 18 we show that both $d(\tau)$ and $\hat{d}(\tau)$ are polynomials of degree $k(k+1)$ and the highest degree coefficient of $\hat{d}(\tau)$ is independent of the sample. The rest of the coefficients of $d(\tau)$ and $\hat{d}(\tau)$ are sums of products of the coefficients of individual entries of the matrices M and \hat{M} respectively.

Note that $\mathbb{E}(\hat{M}) = M$, i.e., for any $1 \leq i, j, \leq (k+1)$, $\mathbb{E}(\hat{M}_{i,j}(\tau)) = M_{i,j}(\tau)$. Since $M_{i,j}(\tau)$ is a polynomial in τ , using standard concentration results we can show that coefficients of the polynomial $\hat{M}_{i,j}(\tau)$ are close to the corresponding coefficients of the polynomial $M_{i,j}(\tau)$ given large enough sample size. Specifically, we show in Lemma 21 that given a sample of size $O\left(\frac{n^{\text{poly}(k)}}{\epsilon^2 \delta}\right)$ each of the coefficients of each of the polynomials $M_{i,j}(\tau)$ can be estimated within error $O\left(\frac{\epsilon}{n^{\text{poly}(k)}}\right)$ with probability at least $1 - \delta$.

Next, in Lemma 22 we show that estimating each of the coefficients of the polynomial $M_{i,j}(\tau)$ for all i, j with accuracy $O\left(\frac{\epsilon}{n^{\text{poly}(k)}}\right)$ ensures that all coefficients of $d(\hat{\tau})$ are $O\left(\frac{\epsilon}{k}\right)$ close to the corresponding coefficients of $d(\tau)$ with high probability.

Consequently, in Lemma 20 we show that when all coefficients of $\hat{d}(\tau)$ are within $O\left(\frac{\epsilon}{k}\right)$ of the corresponding coefficients of $d(\tau)$, the smallest positive root of $\hat{d}(\tau)$, $\hat{\sigma}$, is at most ϵ away from the smallest positive root σ of $d(\tau)$.

Observing that there exist many efficient numerical methods for estimating roots of polynomial of one variable within the desired accuracy completes the proof. ■

References

Achlioptas, D., & McSherry, F. (2005). On Spectral Learning of Mixture of Distributions. *The 18th Annual Conference on Learning Theory*.

- Arora, S., & Kannan, R. (2001). Learning Mixtures of Arbitrary Gaussians. *33rd ACM Symposium on Theory of Computing*.
- Brubaker, S. C., & Vempala, S. (2008). Isotropic pca and affine-invariant clustering. *49th Annual Symposium on Foundations of Computer Science*.
- Burden, R. L., & Faires, J. D. (1993). *Numerical Analysis*. Harcourt Trade Publisher.
- Chaudhuri, K., & Rao, S. (2008a). Beyond Gaussians: Spectral Methods for Learning Mixtures of Heavy Tailed Distributions. *The 21st Annual Conference on Learning Theory*.
- Chaudhuri, K., & Rao, S. (2008b). Learning Mixtures of Product Distributions Using Correlations and Independence. *The 21st Annual Conference on Learning Theory*.
- Chen, H. (1995). Optimal Rate of Convergence for Finite Mixture Models. *The Annals of Statistics*, 23(1), 221–233.
- Dasgupta, A., Hopcroft, J. E., Kleinberg, J., & Sandler, M. (2005). On Learning Mixture of Heavy Tailed Distributions. *46th Annual Symposium on Foundations of Computer Science*.
- Dasgupta, S. (1999). Learning Mixture of Gaussians. *40th Annual Symposium on Foundations of Computer Science*.
- Dasgupta, S., & Schulman, L. (2000). A Two Round Variant of EM for Gaussian Mixtures. *16th Conference on Uncertainty in Artificial Intelligence*.
- Everitt, B. S., & Hand, D. J. (1981). *Finite mixture distributions*. Chapman & Hall.
- Feldman, J., D’Onnell, O., & Servedio, R. (2008). Learning Mixture of Product Distributions over Discrete Domains. *SIAM Journal on Computing*, 37(5), 1536–1564.
- Feldman, J., Servedio, R. A., & O’Donnell, R. (2006). PAC Learning Mixture of Axis Aligned Gaussians with No Separation Assumption. *The 19th Annual Conference on Learning Theory*.
- Kalai, A. T., Moitra, A., & Valiant, G. (2010). Efficiently Learning Mixture of Two Gaussians. *42nd ACM Symposium on Theory of Computing*.
- Kannan, R., Salmasian, H., & Vempala, S. (2005). The Spectral Method for General Mixture Models. *The 18th Annual Conference on Learning Theory*.
- Lindsay, B. G. (1989). Moment Matrices: Applications in Mixtures. *The Annals of Statistics*, 17(2), 722–740.
- Lindsay, B. G. (1995). *Mixture Models: Theory Geometry and Applications*. Institute of Mathematical Statistics.
- McLachlan, G. J., & Basford, K. E. (1988). *Mixture models: Inference and applications to clustering*. Marcel Dekker.
- McLachlan, G. J., & Peel, D. (2000). *Finite mixture models*. John Wiley & Sons.
- Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Phil. Trans. Royal Soc.*, 71110.
- Titterton, D., Smith, A., & Makov, U. (1985). *Statistical analysis of finite mixture distributions*. John Wiley & Sons.
- Vempala, S., & Wang, G. (2002). A Spectral Algorithm for Learning Mixture of Distributions. *43rd Annual Symposium on Foundations of Computer Science*.

A Appendix

In this appendix we provide some of the auxiliary lemmas that are required in the main text. Due to space limitation the proofs are omitted. Extended version of this paper containing the proofs is available at <http://arxiv.org/abs/0907.1054>.

For the purpose of estimating the sample size requirement for an appropriate non-parametric density estimator that is arbitrarily close in L^2 norm sense, we define the Sobolev class as follows. In the following, Sobolev space $W^{2,2}$ is defined as the subset of L^2 such that if $f \in W^{2,2}$ then f and its weak derivatives up to order 2 have finite L^2 norm.

Definition 1 Let $L > 0$. The Sobolev class $\mathcal{S}(2, L)$ is defined as the set of all functions $f : \mathbb{R}^d \rightarrow \mathbb{R}$ such that $f \in W^{2,2}$, and all the second order partial derivatives $\frac{\partial^2 f}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}$, where $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_d)$ is a multi-index with $|\alpha| = 2$, satisfy

$$\left\| \frac{\partial^2 f}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}} \right\|_2 \leq L$$

Note that when the parameters are bounded, mixture of k Gaussian distributions belongs to Sobolev class as defined above and the following Lemma shows that we can approximate the density of such a mixture arbitrarily well in L^2 norm sense.

Lemma 9 Let $p \in \mathcal{S}(2, L)$ be a d -dimensional probability density function and $K : \mathbb{R}^d \rightarrow \mathbb{R}$ be any kernel function with diagonal bandwidth matrix $h^2 \mathbf{I}$, $h > 0$, satisfying $\int K(\mathbf{x}) d\mathbf{x} = 1$, $\int \mathbf{x} K(\mathbf{x}) d\mathbf{x} = \mathbf{0}$, $\int \mathbf{x}^T \mathbf{x} K(\mathbf{x}) d\mathbf{x} < C_1$ and $\int K^2(\mathbf{x}) d\mathbf{x} < C_2$ for positive C_1, C_2 . Then for any $\epsilon_0 > 0$ and any $\delta \in (0, 1)$, with probability greater than $1 - \delta$, the kernel density estimate \hat{p}_S obtained using a sample S of size $\Omega \left(\left[\frac{\log(1/\delta)}{\epsilon_0^2} \right]^d \right)$ satisfies, $\int (p(\mathbf{x}) - \hat{p}_S(\mathbf{x}))^2 d\mathbf{x} \leq \epsilon_0$.

Lemma 10 Consider any set of k points $\{\mathbf{x}_i\}_{i=1}^k$ in \mathbb{R}^d . There exists a direction $\mathbf{v} \in \mathbb{R}^d$, $\|\mathbf{v}\| = 1$ such for any i, j $|\langle \mathbf{x}_i, \mathbf{v} \rangle - \langle \mathbf{x}_j, \mathbf{v} \rangle| > \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{k^2}$.

Note that in the above Lemma dimensionality of the space \mathbb{R}^d is irrelevant but the number of samples k is important. This Lemma can also be considered as a special kind of one sided version of Johnson-Lindenstraus Lemma, and by choosing \mathbf{v} at random from \mathbb{R}^d the same result can be shown to hold with high probability. However, the above result is deterministic.

Lemma 11 Let $g : \mathbb{R}^k \rightarrow \mathbb{R}$ be a continuous bounded function. Let $\mathbf{v}, \mathbf{u}_1, \dots, \mathbf{u}_{k-1} \in \mathbb{R}^k$ be an orthonormal basis of \mathbb{R}^k and let $g_1 : \mathbb{R} \rightarrow \mathbb{R}$ be defined as $g_1(v) = \int \dots \int g(v, u_1, \dots, u_{k-1}) du_1 \dots du_{k-1}$. Then for some $c > 0$, $\|g\|^2 \geq \left(\frac{1}{c\sigma}\right)^k \|g_1\|^2$.

A version of the following Lemma was proved in (Vempala & Wang, 2002). We tailor it for our purpose.

Lemma 12 Let the rows of $A \in \mathbb{R}^{N \times n}$ be picked according to a mixture of Gaussians with means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k \in \mathbb{R}^n$, common variance σ^2 and mixing weights $\alpha_1, \alpha_2, \dots, \alpha_k$ with minimum mixing weight being α_{\min} . Let $\tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_2, \dots, \tilde{\boldsymbol{\mu}}_k$ be the projections of these means on to the subspace spanned by the top k right singular vectors of the sample matrix A . Then for any $0 < \epsilon < 1, 0 < \delta < 1$, with probability at least $1 - \delta$, $\|\boldsymbol{\mu}_i - \tilde{\boldsymbol{\mu}}_i\| \leq \frac{\epsilon}{2}$, provided $N = \Omega \left(\frac{n^3 \sigma^4}{\alpha_{\min}^3 \epsilon^4} \left(\log \left(\frac{n\sigma}{\epsilon \alpha_{\min}} \right) + \frac{1}{n(n-k)} \log \left(\frac{1}{\delta} \right) \right) \right)$,

In the following Lemma we consider a mixture of Gaussians where the mixing weights are allowed to take negative values. This might sound counter intuitive since mixture of Gaussians are never allowed to take negative mixing weights. However, if we have two separate mixtures, for example, one true mixture density $p(\mathbf{x})$ and one its estimate $\hat{p}(\mathbf{x})$, the function $(p - \hat{p})(\mathbf{x})$ that describes the difference between the two densities can be thought of as a mixture of Gaussians with negative coefficients. Our goal is to find a bound of the L^2 norm of such a function.

Lemma 13 Consider a mixture of m k -dimensional Gaussians $f(\mathbf{x}) = \sum_{i=1}^m \alpha_i K(\mathbf{x}, \boldsymbol{\nu}_i)$ where the mixing coefficients $\alpha_i \in (-1, 1)$, $i = 1, 2, \dots, m$. Then the L^2 norm of f satisfies $\|f\|^2 \leq \left(\frac{1}{(2\pi\sigma^2)^k} \right) \boldsymbol{\alpha}^T \hat{\mathbf{K}} \boldsymbol{\alpha}$, where $\hat{\mathbf{K}}$ is a $m \times m$ matrix with $\hat{\mathbf{K}}_{ij} = \exp \left(-\frac{\|\boldsymbol{\nu}_i - \boldsymbol{\nu}_j\|^2}{2\sigma^2} \right)$ and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_m)^T$.

Lemma 14 Let $h : \mathbb{R} \rightarrow \mathbb{C}$ be an infinitely differentiable function such that for some positive integer n and real $M, T > 0$, $|h^{(n)}(0)| > M$ and $|h^{(n+1)}| < T$. Then for any $0 < a < \frac{M}{T\sqrt{2}}$, $|h^{(n)}| > M - \sqrt{2}Ta$ in the interval $[0, a]$.

Lemma 15 Let $h : \mathbb{R} \rightarrow \mathbb{R}$ be an infinitely differentiable function such that for some positive integer n and real $M > 0$, $|h^{(n)}| > M$ in an interval (a, b) . Then $|h^{(n-1)}| > M(b-a)/6$ in a smaller interval either in $(a, \frac{2a+b}{3})$ or in $(\frac{a+2b}{3}, b)$.

Let A be a $k \times k$ Vandermonde matrix defined as follows.

$$A = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_k \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_k^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_1^{k-1} & x_2^{k-1} & x_3^{k-1} & \cdots & x_k^{k-1} \end{bmatrix}$$

Then we can prove the following result.

Lemma 16 For any integer $k > 1$, and positive $a, t \in \mathbb{R}$, let $x_1, x_2, \dots, x_k \in [-a, a]$ and there exists an x_i such that $t = \min_{j, j \neq i} |x_i - x_j|$. Let $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_k) \in \mathbb{R}^k$ with $\min_i |\alpha_i| \geq \alpha_{\min}$. Then for A as defined above, $\|A\alpha\| \geq \alpha_{\min} \left(\frac{t}{1+a}\right)^{k-1}$.

Lemma 17 Consider the $(k+1) \times (k+1)$ Hankel matrix Γ , $\Gamma_{ij} = (\gamma_{i+j}(x, \tau))$ for $i, j = 0, 1, \dots, k$, where $\gamma_n(x, \tau)$ is the n^{th} Hermite polynomial as described above. Then $\det(\Gamma)(x, \tau)$ is a homogeneous polynomial of degree $k(k+1)$ of two variables x and τ .

Using the above Lemma, we have the following simple corollary.

Corollary 18 $d(\tau)$ is a polynomial of degree $k(k+1)$, with the coefficient of the leading term independent of the probability distribution p . Similarly, $\hat{d}(\tau)$ is a polynomial of degree $k(k+1)$, with the leading term having coefficient independent of the coefficients of the sampled data.

Lemma 19 Let $f(x) = x^m + a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0$ be a polynomial having a smallest positive real root x_0 with multiplicity one and $f'(x_0) \neq 0$. Let $\hat{f}(x) = x^m + \hat{a}_{m-1}x^{m-1} + \hat{a}_{m-2}x^{m-2} + \cdots + \hat{a}_1x + \hat{a}_0$ be another polynomial such that $\|a - \hat{a}\| \leq \epsilon$ for some sufficiently small $\epsilon > 0$, where $a = (a_0, a_1, \dots, a_{m-1})$ and $\hat{a} = (\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{m-1})$. Then there exists a $C > 0$ such that the smallest positive root \hat{x}_0 of $\hat{f}(x)$ satisfies $\|x_0 - \hat{x}_0\| \leq C\epsilon$.

Lemma 20 Let σ be the smallest positive root of $d(\tau)$. Suppose $\hat{d}(\tau)$ be the polynomial where each of the coefficients of $d(\tau)$ are estimated within ϵ error for some sufficiently small $\epsilon > 0$. Let $\hat{\sigma}$ be the smallest positive root of $\hat{d}(\tau)$. Then $|\hat{\sigma} - \sigma| = O(k\epsilon)$.

The matrix M defined in Section 3.3 has $2k$ different entries which is clear from its construction. Each such entry is a polynomial in τ . Let us denote these distinct entries by $m_i(\tau) = \mathbb{E}[\gamma_i(x, \tau)]$, $i = 1, 2, \dots, 2k$. Note that the empirical version of the matrix M is \hat{M} where each entry $m_i(\tau)$ is replaced by its empirical counterpart $\hat{m}_i(\tau)$. Using standard concentration inequality we show that for any $m_i(\tau)$, its coefficients are arbitrarily close to the corresponding coefficients of $\hat{m}_i(\tau)$ provided a large enough sample size is used to estimate $\hat{m}_i(\tau)$.

Lemma 21 For any $m_i(\tau)$, $i = 1, 2, \dots, 2k$, let β be any arbitrary coefficient of the polynomial $m_i(\tau)$. Suppose X_1, X_2, \dots, X_N iid samples from p is used to estimate $\hat{m}_i(\tau)$ and the corresponding coefficient is $\hat{\beta}$. Then there exists a polynomial $\eta_1(k)$ such that for any $\epsilon > 0$ and $0 < \delta$, $|\beta - \hat{\beta}| \leq \epsilon$ with probability at least $1 - \delta$, provided $N > \frac{n^{\eta_1(k)}}{\epsilon^2 \delta}$.

Lemma 22 There exists a polynomial $\eta_2(k)$ such that if coefficients of each of the entries of matrix M (where each such entry is a polynomial of τ) are estimated within error $\frac{\epsilon}{n^{\eta_2(k)}}$ then each of the coefficients of $d(\tau)$ are estimated within ϵ error.

Sparse Recovery in Convex Hulls of Infinite Dictionaries

Vladimir Koltchinskii
Georgia Institute of Technology
vlad@math.gatech.edu

Stas Minsker
Georgia Institute of Technology
sminsker@math.gatech.edu

Abstract

Let S be an arbitrary measurable space, $T \subset \mathbb{R}$ and (X, Y) be a random couple in $S \times T$ with unknown distribution P . Let $(X_1, Y_1), \dots, (X_n, Y_n)$ be i.i.d. copies of (X, Y) . Denote by P_n the empirical distribution based on the sample $(X_i, Y_i), i = 1, \dots, n$. Let \mathcal{H} be a set of uniformly bounded functions on S . Suppose that \mathcal{H} is equipped with a σ -algebra and with a finite measure μ . Let \mathbb{D} be a convex set of probability densities with respect to μ . For $\lambda \in \mathbb{D}$, define the mixture $f_\lambda(\cdot) = \int h(\cdot)\lambda(h)d\mu(h)$. Given a loss function $\ell : T \times \mathbb{R} \mapsto \mathbb{R}$ such that, for all $y \in T$, $\ell(y, \cdot)$ is convex, denote $(\ell \bullet f)(x, y) = \ell(y; f(x))$. We study the following penalized empirical risk minimization problem

$$\hat{\lambda}_\varepsilon := \operatorname{argmin}_{\lambda \in \mathbb{D}} \left[P_n(\ell \bullet f_\lambda) + \varepsilon \int \lambda \log \lambda d\mu \right]$$

along with its distribution dependent version

$$\lambda_\varepsilon := \operatorname{argmin}_{\lambda \in \mathbb{D}} \left[P(\ell \bullet f_\lambda) + \varepsilon \int \lambda \log \lambda d\mu \right].$$

We prove that the “approximate sparsity” of λ_ε implies the “approximate sparsity” of $\hat{\lambda}_\varepsilon$ and study connections between the sparsity and the excess risk of empirical solutions $\hat{\lambda}_\varepsilon$.

1 Introduction

Sparsity phenomena in empirical risk minimization over linear spans or convex hulls of large finite dictionaries have been extensively studied in the recent years (see, e.g., [MPTJ07], [Kol09b], [BRT09] and references therein). In this paper, our goal is to extend some of these results to the case of empirical risk minimization over convex hulls of infinite dictionaries which is a standard framework in machine learning (for instance, in large margin classification, the dictionaries are often infinite families of functions such as decision stumps, decision trees or subsets of reproducing kernel Hilbert spaces).

Let S be a measurable space, $T \subset \mathbb{R}$ be a Borel set and (X, Y) be a random couple in $S \times T$ with unknown distribution P . The marginal distribution of X will be denoted by Π . Let $(X_1, Y_1), \dots, (X_n, Y_n)$ be the training data consisting of n i.i.d. copies of (X, Y) . In what follows, we will denote by P_n the empirical

*Partially supported by NSF grants MSPA-MCS-0624841, DMS-0906880 and CCF-0808863

distribution based on a given sample of n training examples. Similarly, Π_n will denote the empirical measure based on the sample (X_1, \dots, X_n) . The integrals with respect to P and P_n are denoted by

$$Pg := \mathbb{E}g(X, Y), \quad P_n g := \frac{1}{n} \sum_{i=1}^n g(X_i, Y_i).$$

Similar notations will be used for Π, Π_n and other measures. Let $\ell(y, \cdot)$ be the loss function such that for all $y \in T$, $\ell(y, \cdot)$ is convex. For a function $f : S \mapsto \mathbb{R}$, let $(\ell \bullet f)(x, y) := \ell(y, f(x))$. A *dictionary* is a given family \mathcal{H} of measurable functions $h : S \mapsto [-1, 1]$. Assume that \mathcal{H} is equipped with a σ -algebra and with a finite measure μ . In what follows, the complexity of the dictionary \mathcal{H} will be characterized in terms of $L_2(\Pi)$ and $L_2(\Pi_n)$ covering numbers. Suppose Λ is a probability measure on \mathcal{H} absolutely continuous with respect to μ with $\lambda = \frac{d\Lambda}{d\mu}$. The (negative) entropy $H(\lambda)$ is defined as $H(\lambda) := \int_{\mathcal{H}} \lambda(h) \log \lambda(h) d\mu(h)$. In what follows, we consider only densities with finite entropies. Let f_λ denote the mixture of the functions from the dictionary \mathcal{H} with respect to $\lambda : f_\lambda(\cdot) := \int_{\mathcal{H}} h(\cdot) \lambda(h) \mu(dh)$. The excess risk $\mathcal{E}(f)$ of a function f is defined as

$$\mathcal{E}(f) = P(\ell \bullet f) - \inf_{g: S \mapsto \mathbb{R}} P(\ell \bullet g) = P(\ell \bullet f) - P(\ell \bullet f_*).$$

For simplicity, we assume throughout the paper that $\inf_{g: S \mapsto \mathbb{R}} P(\ell \bullet g)$ is attained at some uniformly bounded function f_* (the infimum is taken over all measurable functions).

Let \mathbb{D} be a convex set of probability densities on \mathcal{H} . We will assume that, for all $\lambda \in \mathbb{D}$, $\lambda \log \lambda \in L_1(\mu)$, so, the entropy $H(\lambda)$ is finite. Consider the following penalized risk minimization problem

$$\lambda_\varepsilon := \operatorname{argmin}_{\lambda \in \mathbb{D}} F(\lambda), \quad F(\lambda) := P(\ell \bullet f_\lambda) + \varepsilon H(\lambda) \quad (1.1)$$

together with its empirical version:

$$\hat{\lambda}_\varepsilon := \operatorname{argmin}_{\lambda \in \mathbb{D}} F_n(\lambda), \quad F_n(\lambda) := P_n(\ell \bullet f_\lambda) + \varepsilon H(\lambda). \quad (1.2)$$

Note that, due to the convexity of the loss, of the negative entropy and of the set \mathbb{D} , both (1.1) and (1.2) are convex optimization problems. We will use the notations $\Lambda_\varepsilon, \hat{\Lambda}_\varepsilon$ for the probability measures with densities $\lambda_\varepsilon, \hat{\lambda}_\varepsilon$, respectively.

Our first aim is to study problem (1.1) and to bound the approximation error $\mathcal{E}(f_{\lambda_\varepsilon})$ of its solution, which, for the losses of quadratic type, is equivalent to bounding the $L_2(\Pi)$ -approximation error $\|f_{\lambda_\varepsilon} - f_*\|_{L_2(\Pi)}^2$. We show that the size of this error can be controlled in terms of the approximation error of oracle solutions $\lambda \in \mathbb{D}$ that are “sparse” in the sense that they are concentrated on a “small” set of functions $\mathcal{H}' \subset \mathcal{H}$ and, at the same time, possess some regularity properties. Moreover, we show that if there exist “sparse” oracles providing good approximation of the target function, then solutions λ_ε of problem (1.1) are “approximately sparse” in the sense that they “concentrate” on the support of “sparse” oracles.

Next, we study the relationship between the problems (1.1) and (1.2). We show that the “approximate sparsity” of the true penalized solution λ_ε implies that the corresponding empirical solution $\hat{\lambda}_\varepsilon$ possesses the same property with a high probability. More precisely, if there exists a measurable set $\mathcal{H}' \subset \mathcal{H}$ such that $\Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}')$ is small and, at the same time, there exists a subspace $L \subset L_2(\Pi)$ of small dimension d that provides a good $L_2(\Pi)$ -approximation of the functions from the set \mathcal{H}' , we will show that in this case, with a high probability, the empirical solution $\hat{\lambda}_\varepsilon$ is also approximately supported on the same set \mathcal{H}' in the sense that $\hat{\Lambda}_\varepsilon(\mathcal{H} \setminus \mathcal{H}')$ is small. Thus, both the empirical solution $\hat{\lambda}_\varepsilon$ and the true solution λ_ε follow the same “sparsity pattern”: they are concentrated on the same set of functions \mathcal{H}' which can be well approximated by a linear

subspace of small dimension. We also obtain probabilistic bounds on the random error $|\mathcal{E}(f_{\hat{\lambda}_\varepsilon}) - \mathcal{E}(f_{\lambda_\varepsilon})|$, or, equivalently, the $L_2(\Pi)$ -random error $\|f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}\|_{L_2(\Pi)}^2$, in terms of characteristics of the sparsity of the problem such as the measure $\Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}')$ and the dimension d of the approximating space L . At the same time, we derive upper bounds on the Kullback-Leibler type distance between $\hat{\lambda}_\varepsilon$ and λ_ε .

The idea of using entropy for complexity penalization is not new in machine learning (see, e.g., [Zha01]). An approach to sparse recovery based on entropy penalization has been studied in the case of finite dictionaries \mathcal{H} (see [Kol09a], [Kol08]). As in these papers, the fact that the penalty is strictly convex allows us to study the random error independently of the approximation error, but geometric parameters of the dictionary needed to control these two errors are not quite the same. ℓ_1 -type penalization in the case of infinite dictionaries was suggested in [RSSZ07] (however, sharp generalization error bounds were not studied in this paper).

2 Preliminaries

Assumptions on the loss. Assume that for all $y \in T$, $\ell(y, \cdot)$ is a convex twice differentiable function, ℓ''_u [here and in what follows the derivatives of the loss are taken with respect to the second variable] is uniformly bounded in $T \times [-1, 1]$ and $\sup_{y \in T} \ell(y; 0) < +\infty$, $\sup_{y \in T} |\ell'_u(y; 0)| < +\infty$. It will be also assumed that

$$m := \frac{1}{2} \inf_{y \in T} \inf_{|u| \leq 1} \ell''_u(y, u) > 0.$$

In what follows, the loss functions ℓ satisfying the above assumptions will be called **the losses of quadratic type**. In particular, the assumptions imply that

$$m \|f_\lambda - f_*\|_{L_2(\Pi)}^2 \leq \mathcal{E}(f_\lambda) \leq M \|f_\lambda - f_*\|_{L_2(\Pi)}^2,$$

where $M := \frac{1}{2} \sup_{y, u} \ell''_u(y, u)$. Moreover, the following simple proposition also holds for such losses:

Proposition 1 *There exists a constant $C > 0$ depending only on ℓ such that for all $\lambda, \bar{\lambda} \in \mathbb{D}$,*

$$|\mathcal{E}(f_{\bar{\lambda}}) - \mathcal{E}(f_\lambda)| \leq C \left[\|f_{\bar{\lambda}} - f_\lambda\|_{L_2(\Pi)}^2 \sqrt{\mathcal{E}(f_\lambda)} \|f_{\bar{\lambda}} - f_\lambda\|_{L_2(\Pi)} \right].$$

Proof: See the proof of Theorem 3 in [Kol09a]. □

Common examples of such loss functions include the usual quadratic loss $\ell(y, u) = (y - u)^2$ used in the regression setting (with T being a bounded interval of \mathbb{R}) as well as the exponential loss $\ell(y, u) = e^{-yu}$ and the logit loss $\ell(y, u) = \log_2(1 + e^{-yu})$ used in large margin classification (with $T = \{-1, 1\}$).

Existence of solutions. We provide sufficient conditions of existence of solutions of problems (1.1) and (1.2). Recall that all the densities λ in question have finite entropy.

Proposition 2 *Problems (1.1), (1.2) have unique solutions in every convex weakly compact subset \mathbb{D} of L_p , $p \geq 1$.*

Proof: First, we show that the entropy functional $H(\lambda) := \int_{\mathcal{H}} \lambda \log \lambda d\mu$ is lower semi-continuous in $L_p(\mu)$, $p \geq 1$. Indeed, the functional is lower semi-continuous iff the level sets $\mathcal{L}_t = \{\lambda : H(\lambda) \leq t\}$ are closed. Suppose $\lambda_n \in \mathcal{L}_t$, $\lambda_n \rightarrow \lambda_0$ in L_p . We can extract the subsequence λ_{n_k} converging to λ_0 pointwise. Noting that $s \log(s) + e^{-1} \geq 0$ and applying the Fatou lemma to the sequence $\{\lambda_{n_k} \log(\lambda_{n_k})\}$, we derive

the result. Next, under the assumptions on the loss, $F(\lambda)$ is convex, bounded from below and lower semi-continuous (continuity of the risk $P(\ell \bullet f_\lambda)$ follows from the uniform boundedness of the dictionary and integral Minkowski inequality), so that the level sets $\mathcal{L}_t = \{\lambda : F(\lambda) \leq t\}$ are closed and convex. Mazur's theorem implies that they are also closed in weak topology, so F is weakly lower semi-continuous. Given a minimizing sequence λ_n , we can extract a weakly convergent subsequence $\lambda_{n_k} \rightarrow \lambda_\infty$, and conclude that $\lambda_\infty \in \mathbb{D}$, $-\infty < F(\lambda_\infty) \leq \liminf_{k \rightarrow \infty} F(\lambda_{n_k})$. Convexity of the set \mathbb{D} and strict convexity of the functional F implies the uniqueness of the solution of (1.1). Replacing F by F_n , we get similar statements for (1.2). \square

We will assume throughout the paper that \mathbb{D} is a convex set of probability densities such that $\lambda \log \lambda \in L_1(\mu)$, $\lambda \in \mathbb{D}$ and solutions of the problems (1.1), (1.2) exist in \mathbb{D} .

Differentiability of the risk and of the entropy. To derive necessary conditions of the minima in the optimization problems (1.1), (1.2), we have to study differentiability properties of the functions involved. For $G : \mathbb{D} \mapsto \mathbb{R}$, $\lambda \in \mathbb{D}$ and ν such that $\bar{\lambda} := \lambda + t_0\nu \in \mathbb{D}$ for some $t_0 > 0$, denote

$$DG(\lambda; \nu) := \lim_{t \downarrow 0} \frac{G(\lambda + t\nu) - G(\lambda)}{t},$$

provided that the limit exists. $DG(\lambda; \nu)$ is the (directional) derivative of G at point λ in the direction ν .

First note that, under our assumptions on the loss function ℓ , both the true risk $\mathbb{D} \ni \lambda \mapsto P(\ell \bullet f_\lambda) =: L(\lambda)$ and the empirical risk $\mathbb{D} \ni \lambda \mapsto P(\ell \bullet f_\lambda) =: L_n(\lambda)$ have directional derivatives at any point $\lambda \in \mathbb{D}$ in the direction of any other point $\bar{\lambda} = \lambda + t_0\nu \in \mathbb{D}$, $t_0 > 0$. Moreover, the following formulas hold:

$$DL(\lambda, \nu) = P(\ell' \bullet f_\lambda) f_\nu \text{ and } DL_n(\lambda, \nu) = P_n(\ell' \bullet f_\lambda) f_\nu. \quad (2.1)$$

Let λ_1, λ_2 be two densities from \mathbb{D} and Λ_1, Λ_2 the corresponding probability measures on \mathcal{H} . Denote by $K(\lambda_1|\lambda_2) := \int_{\mathcal{H}} \log \frac{\lambda_1}{\lambda_2} \lambda_1 d\mu$ the Kullback-Leibler divergence between λ_1 and λ_2 and let $K(\lambda_1, \lambda_2) := K(\lambda_1|\lambda_2) + K(\lambda_2|\lambda_1)$ be the symmetrized Kullback-Leibler divergence.

Proposition 3 For all $\lambda_1, \lambda_2 \in \mathbb{D}$, $\tau \in (0, 1)$ and measurable $\mathcal{H}' \subset \mathcal{H}$

$$DH(\lambda_1 + \tau(\lambda_2 - \lambda_1); \lambda_2 - \lambda_1) = \int_{\mathcal{H}} \log(\lambda_1 + \tau(\lambda_2 - \lambda_1))(\lambda_2 - \lambda_1) d\mu, \quad (2.2)$$

$$K(\lambda_1, \lambda_2) = \lim_{t \rightarrow 0} \int_{\mathcal{H}} \log \frac{(1-t)\lambda_1 + t\lambda_2}{t\lambda_1 + (1-t)\lambda_2} (\lambda_1 - \lambda_2) d\mu, \quad (2.3)$$

$$\Lambda_1(\mathcal{H} \setminus \mathcal{H}') \leq 2\Lambda_2(\mathcal{H} \setminus \mathcal{H}') + K(\lambda_1, \lambda_2). \quad (2.4)$$

The proof of (2.2) and (2.3) is based on a convexity argument and on the monotone convergence theorem. To show (2.4), note that by the well known inequality relating the Kullback-Leibler and Hellinger distances, for all $\mathcal{H}' \subset \mathcal{H}$

$$\begin{aligned} K(\lambda_1, \lambda_2) &\geq 2 \int_{\mathcal{H}} \left(\sqrt{\lambda_1} - \sqrt{\lambda_2} \right)^2 \geq 2 \int_{\mathcal{H} \setminus \mathcal{H}'} \left(\sqrt{\lambda_1} - \sqrt{\lambda_2} \right)^2 \geq \\ &\geq 2 \int_{\mathcal{H} \setminus \mathcal{H}'} \left(\lambda_1 + \lambda_2 - \frac{\lambda_1}{2} - 2\lambda_2 \right) = \Lambda_1(\mathcal{H} \setminus \mathcal{H}') - 2\Lambda_2(\mathcal{H} \setminus \mathcal{H}'). \end{aligned}$$

3 Bounding approximation error

In what follows, our goal is to compare the excess risk of the estimator λ_ε with the excess risk of "oracles" $\lambda \in \mathbb{D}$. Define a cone $\mathbb{K} := \{c(\lambda_1 - \lambda_2) : \lambda_1, \lambda_2 \in \mathbb{D}, c \in \mathbb{R}\}$ and for $w \in L_2(\mu)$, define **the alignment**

coefficient $\gamma(w)$ to be

$$\gamma(w) := \sup \{ \langle w, u \rangle_{L_2(\mu)} : \|f_u\|_{L_2(\Pi)} = 1, u \in \mathbb{K} \}.$$

It is easy to see that, for all constants $c \in \mathbb{R}$, $\gamma(w + c) = \gamma(w)$. Denote by K the Gram operator of the dictionary, i.e.,

$$(Ku)(h) = \int_{\mathcal{H}} \langle h, g \rangle_{L_2(\Pi)} u(g) \mu(dg), \quad h \in L_2(\Pi),$$

which is a self-adjoint nonnegatively definite operator. Clearly,

$$\|f_u\|_{L_2(\Pi)}^2 = \langle Ku, u \rangle_{L_2(\mu)} = \langle K^{\frac{1}{2}}u, K^{\frac{1}{2}}u \rangle_{L_2(\mu)}$$

and it is easy to see that for all $w \in \text{Im}(K^{1/2})$, $\gamma(w) \leq \|K^{-\frac{1}{2}}w\|_{L_2(\mu)}$. Roughly speaking, $\gamma(w)$ is small if the function w is “properly aligned” with eigenspaces of the Gram operator K of the dictionary (say, it belongs to the linear span of eigenspaces corresponding to large enough eigenvalues of K).

The following theorem shows that the approximation error $\mathcal{E}(f_{\lambda_\varepsilon})$ of the true solution λ_ε can be controlled by the approximation error $\mathcal{E}(f_\lambda)$ of “oracles” $\lambda \in \mathbb{D}$ up to an error term of the size $\gamma^2(\log \lambda)\varepsilon^2$. Moreover, it also shows that, for any oracle $\lambda \in \mathbb{D}$, f_{λ_ε} belongs to an $L_2(\Pi)$ ball around f_λ whose radius is, up to a constant, $\|f_\lambda - f_*\|_{L_2(\Pi)} \vee \gamma(\log \lambda)\varepsilon$. At the same time, λ_ε belongs to a Kullback-Leibler “ball” around λ whose radius is $\frac{1}{\varepsilon} \|f_\lambda - f_*\|_{L_2(\Pi)}^2 \vee \gamma^2(\log \lambda)\varepsilon$. Thus, the existence of an oracle λ that approximates the target function well (i.e., $\|f_\lambda - f_*\|_{L_2(\Pi)}$ is small) and that is “well aligned” with the dictionary (i.e., $\gamma(\log \lambda)$ is small) would imply that f_{λ_ε} is $L_2(\Pi)$ -close to f_λ and, at the same time, λ_ε is close to λ in the Kullback-Leibler distance. It would also imply that the approximation error $\mathcal{E}(f_{\lambda_\varepsilon})$ is small and that the measures Λ_ε and Λ (with densities λ_ε and λ) have similar “concentration pattern” (as the last two inequalities of the theorem show).

Theorem 1 *There exists a constant $C > 0$ depending only on the loss such that, for all oracles $\lambda \in \mathbb{D}$,*

$$\|f_{\lambda_\varepsilon} - f_\lambda\|_{L_2(\Pi)}^2 + \varepsilon K(\lambda_\varepsilon, \lambda) \leq C \left[\|f_\lambda - f_*\|_{L_2(\Pi)}^2 \vee \gamma^2(\log \lambda)\varepsilon^2 \right].$$

Moreover, the following bound on the excess risk of λ_ε holds

$$\mathcal{E}(f_{\lambda_\varepsilon}) \leq \inf_{\lambda \in \mathbb{D}} \left[\mathcal{E}(f_\lambda) + C \sqrt{\mathcal{E}(f_\lambda)} \gamma(\log \lambda) \varepsilon + C \gamma^2(\log \lambda) \varepsilon^2 \right]$$

and, for all $\mathcal{H}' \subset \mathcal{H}$,

$$\begin{aligned} \Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}') &\leq 2\Lambda(\mathcal{H} \setminus \mathcal{H}') + \frac{C}{\varepsilon} \left[\|f_\lambda - f_*\|_{L_2(\Pi)}^2 \vee \gamma^2(\log \lambda)\varepsilon^2 \right], \\ \Lambda(\mathcal{H} \setminus \mathcal{H}') &\leq 2\Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}') + \frac{C}{\varepsilon} \left[\|f_\lambda - f_*\|_{L_2(\Pi)}^2 \vee \gamma^2(\log \lambda)\varepsilon^2 \right]. \end{aligned}$$

In concrete examples below, the dictionary is of the form $\mathcal{H} = \{h_t : t \in I\}$, where $I \subset \mathbb{R}^d$ is a bounded domain in \mathbb{R}^d . In such cases, one can assume that μ is a measure on I and the mixing densities λ are functions on I . Often, it happens that $K^{-1/2}$ can be defined in terms of certain differential operators and the alignment coefficient $\gamma(w)$ is bounded by a Sobolev type norm of the function w : for some $A > 0$ and $\alpha > 0$,

$$\gamma(w) \leq A \|w\|_{\mathbb{W}^{2,\alpha}(I)}. \quad (3.1)$$

We are interested in those oracles $\lambda \in \Lambda$ for which $\gamma(\log \lambda)$ is not too large and it is controlled by “smoothness” and “sparsity” of λ . Assume that μ is the Lebesgue measure on I and that condition (3.1) holds. Let

$\lambda := \sum_{j=1}^d \lambda_j + \delta$, where $\delta \in (0, 1)$ and λ_j are nonnegative functions, $\lambda_j \in C^\infty(\mathbb{R}^d)$, $\text{supp}(\lambda_j) \subset U_j$, $U_j \subset T$ being disjoint balls. Finally, we assume that $\sum_{j=1}^d \int_{\mathbb{R}^d} \lambda_j(t) dt = 1 - \delta$. Then it is easy to see that

$$\log \lambda = \sum_{j=1}^d (\log(\lambda_j + \delta) - \log \delta) + \log \delta$$

and, for each $j = 1, \dots, d$, the function $w_j := \log(\lambda_j + \delta) - \log \delta \in C^\infty(\mathbb{R}^d)$ and it is supported in U_j . Therefore, since the functions w_j have disjoint supports,

$$\gamma(\log \lambda) \leq A_1 \left\| \sum_{j=1}^d w_j \right\|_{\mathbb{W}^{2,\alpha}(I)} \leq A \left(\sum_{j=1}^d \|w_j\|_{\mathbb{W}^{2,\alpha}(I)}^2 \right)^{1/2}.$$

In this model, δ plays the role of a small “background density” (needed to make λ bounded away from 0) and densities λ_j , $j = 1, \dots, d$ are “spikes”. The resulting oracle density λ is “approximately” sparse in the sense that most of the mass is concentrated in a small part of the space (in the union of balls U_j). For smooth enough “spikes“, $\gamma(\log \lambda)$ becomes of the order \sqrt{d} , so, it depends on the “sparsity“ of the problem.

We now consider three more specific examples of the dictionaries.

Fourier dictionary. Suppose that $S := \mathbb{R}^d$ and let $\mathcal{H} = \{\cos(t, \cdot), t \in I\}$, where $I \subset \mathbb{R}^d$ is a bounded open set symmetric about the origin, i.e., $I = -I$. It can be assumed now that the measure μ and the densities λ are defined on the set I . Suppose that measures μ, Π are absolutely continuous with respect to the Lebesgue measure with densities m and p , respectively. It will be assumed that $m(t) = m(-t)$, $t \in I$. We will also assume that for $\lambda \in \mathbb{D}$, $\lambda(t) = \lambda(-t)$, $t \in I$. When it is needed, it will be assumed that functions λ, m are defined on the whole space \mathbb{R}^d and are equal to 0 on $\mathbb{R}^d \setminus I$. Clearly, the function f_λ is then the Fourier transform of λm :

$$f_\lambda(\cdot) = \int_{\mathbb{R}^d} e^{i\langle t, \cdot \rangle} \lambda(t) m(t) dt := \widehat{\lambda m}(\cdot).$$

Therefore, assuming that the density p is positive, we get, for all $w \in C^\infty(\mathbb{R}^d)$, $u \in \mathbb{K}$

$$\langle w, u \rangle_{L_2(\mu)} = \langle w, u \cdot m \rangle_{L_2(\mathbb{R}^d)} = \langle \widehat{w}, \widehat{u m} \rangle_{L_2(\mathbb{R}^d)} = \langle \widehat{w}, f_u \rangle_{L_2(\mathbb{R}^d)} = \left\langle \frac{\widehat{w}}{p^{1/2}}, f_u p^{1/2} \right\rangle_{L_2(\mathbb{R}^d)},$$

which easily implies that $\gamma(w) \leq \left\| \frac{\widehat{w}}{\sqrt{p}} \right\|_{L_2(\mathbb{R}^d)}$. Under an additional assumption that for some $L > 0$, $\alpha > 0$, $p(x) \geq L(1 + |x|^2)^{-\alpha}$, $x \in \mathbb{R}^d$, we get the following bound: $\gamma(w) \leq A_1 \|(I + \Delta)^{\alpha/2} w\|_{L_2(\mathbb{R}^d)} \leq A \|w\|_{\mathbb{W}^{2,\alpha}(\mathbb{R}^d)}$ (where Δ stands for the Laplace operator).

Location dictionary. Suppose now that $S := \mathbb{T}^d$ is the d -dimensional torus and let $\mathcal{H} = \{h(\cdot - \theta), \theta \in \mathbb{T}^d\}$ for some bounded function $h : \mathbb{T}^d \rightarrow \mathbb{R}$ and let μ be the Haar measure on \mathbb{T}^d . Assume that Π is a probability measure on \mathbb{T}^d with density p (with respect to the Haar measure) that is bounded away from 0 by a constant $L > 0$. Then, a simple Fourier analysis argument shows that

$$\gamma(w) \leq A \left(\sum_{n \in \mathbb{Z}^d} \left| \frac{\widehat{w}_n}{\widehat{h}_n} \right|^2 \right)^{1/2},$$

where $\widehat{w}_n, \widehat{h}_n$ denote the Fourier coefficients of functions w, h . Under the assumption that $|\widehat{h}_n| \geq L(1 + |n|^2)^{-\alpha/2}$, it easily follows that

$$\gamma(w) \leq A \|w\|_{\mathbb{W}^{2,\alpha}(\mathbb{T}^d)}.$$

Monotone functions dictionary. Assuming that $S = [0, 1]$, let $\mathcal{H} := \{I_{[0,s]} : s \in [0, 1]\}$ and let μ be the Lebesgue measure in $[0, 1]$. The mixtures of functions from \mathcal{H} are decreasing absolutely continuous functions $f : [0, 1] \mapsto [0, 1]$ such that $f(0) = 1$ and $f(1) = 0$. Suppose that Π is the Lebesgue measure in $[0, 1]$. The Gram operator K is given by the kernel $K(s, t) = \langle I_{[0,s]}, I_{[0,t]} \rangle_{L_2(\Pi)} = \min(s, t)$. Clearly, K is a compact self-adjoint operator. It is well known that its eigenvalues are $\left(\frac{1}{\pi(k+1/2)}\right)^2$ and the corresponding eigenfunctions are $\phi_k(t) = \sqrt{2} \sin((k+1/2)\pi t)$, $k = 0, 1, 2, \dots$. For a function $w \in \mathbb{W}^{2,1}[0, 1]$, $w(0) = 0$, $w = \sum_{k=0}^{\infty} w_k \phi_k$, we have

$$\left(K^{-1/2}w\right)(t) = \sum_{k=0}^{\infty} \pi(k+1/2)w_k \phi_k(t) = w'(t).$$

Hence

$$\gamma(w) \leq \|K^{-1/2}w\|_{L_2[0,1]} = \pi \left(\sum_{k=0}^{\infty} (k+1/2)^2 w_k^2 \right)^{1/2} \leq A \|w\|_{\mathbb{W}^{2,1}[0,1]}.$$

Assume again that \mathcal{H} is an arbitrary dictionary.

Weakly correlated partitions. Let $\mathcal{H}_j, j = 1, \dots, N$ be a measurable partition of \mathcal{H} . As a concrete example of such a partition, one can consider the case when $S = [0, 1]^N$ and, for each $j = 1, \dots, N$, \mathcal{H}_j is a class of functions depending on the j -th variable. We are interested in the situation when the number N of function classes \mathcal{H}_j is large and they are "weakly correlated". This might be viewed as an extension to the case of infinite dictionaries of usual notions of "almost orthogonality" (such as, for instance, restricted isometry property of Candes and Tao) frequently used in the literature on sparse recovery. It is also close to "sparse additive models" and "sparse multiple kernel learning" (see [KY08], [MvdGB09]). Suppose there exist oracles $\lambda \in \mathbb{D}$ such that f_λ provides a good approximation of the target f_* and, at the same time, λ is "sparse" in the sense that it is concentrated mostly on a small number of sets \mathcal{H}_j . For each set \mathcal{H}_j , let $K_j : L_2(\mathcal{H}_j; \mu) \mapsto L_2(\mathcal{H}_j, \mu)$ be the integral operator (self-adjoint and nonnegatively definite) defined by

$$(K_j u)(h) := \int_{\mathcal{H}_j} \text{cov}_\Pi(h, g) u(g) \mu(dg), \quad h \in \mathcal{H}_j,$$

where $\text{cov}_\Pi(h, g) := \Pi(hg) - \Pi(h)\Pi(g)$. We will also denote

$$\sigma_\Pi(g) := \sqrt{\text{cov}_\Pi(g, g)} \quad \text{and} \quad \rho_\Pi(h, g) := \frac{\text{cov}_\Pi(h, g)}{\sigma_\Pi(h)\sigma_\Pi(g)}.$$

Let \mathcal{L}_j be the subspace of $L_2(\Pi)$ spanned by \mathcal{H}_j and, for $J \subset \{1, \dots, N\}$, let

$$\beta_2(J) := \inf \left\{ \beta > 0 : \forall f_j \in \mathcal{L}_j, j = 1, \dots, N \quad \sum_{j \in J} \sigma_\Pi^2(f_j) \leq \beta^2 \sigma_\Pi^2 \left(\sum_{j=1}^N f_j \right) \right\}.$$

Note that if the spaces $\mathcal{L}_j, j = 1, \dots, N$ are uncorrelated, i.e., $\text{cov}_\Pi(h, g) = 0, h \in \mathcal{L}_i, g \in \mathcal{L}_j, i \neq j$, then $\beta_2(J) = 1$. More generally, given $h_j \in \mathcal{L}_j, j = 1, \dots, N$, denote by $\kappa(\{h_j : j \in J\})$ the minimal eigenvalue of the covariance matrix $(\text{cov}_\Pi(h_i, h_j))_{i,j \in J}$. Let

$$\kappa(J) := \inf \left\{ \kappa(\{h_j : j \in J\}) : h_j \in \mathcal{L}_j, \sigma_\Pi(h_j) = 1 \right\}.$$

Denote $\mathcal{L}_J = \text{l.s.} \left(\bigcup_{j \in J} \mathcal{L}_j \right)$ (here l.s. means linear span) and let $\rho(J) := \sup \left\{ \rho_\Pi(f, g) : f \in \mathcal{L}_J, g \in \mathcal{L}_{J^c} \right\}$. The quantity $\rho(J)$ should be compared with the notion of **canonical correlation** often used in the multivariate statistical analysis. It is easy to check (see [Kol08], proposition 7.1) that

$$\beta_2(J) \leq \frac{1}{\sqrt{\kappa(J)(1 - \rho^2(J))}}.$$

The next proposition easily follows from the definitions of $\gamma(w), \beta_2(J)$ and the operators K_j .

Proposition 4 For all $J \subset \{1, \dots, N\}$ and all $w = \sum_{j \in J} w_j$ with $w_j \in \text{Im}(K_j^{1/2})$,

$$\gamma(w) \leq \beta_2(J) \left(\sum_{j \in J} \|K_j^{-1/2} w_j\|_{L_2(\mathcal{H}_j, \mu)}^2 \right)^{1/2}. \quad (3.2)$$

If now $\lambda := \sum_{j \in J} \lambda_j + \delta$, where $\delta \in (0, 1)$, λ_j are nonnegative functions defined on \mathcal{H}_j and

$$\sum_{j=1}^d \int_{\mathcal{H}_j} \lambda_j(h) dh = 1 - \delta,$$

then $\log \lambda = \sum_{j \in J} w_j I_{\mathcal{H}_j} + \log \delta$, where $w_j := \log(\lambda_j + \delta) - \log \delta$. Therefore, (3.2) implies

$$\gamma(\log \lambda) \leq \beta_2(J) \left(\sum_{j \in J} \|K_j^{-1/2} w_j\|_{L_2(\mathcal{H}_j, \mu)}^2 \right)^{1/2}.$$

4 Bounding Random Error

The purpose of this section is to develop exponential bounds on the random error $|\mathcal{E}(f_{\hat{\lambda}_\varepsilon}) - \mathcal{E}(f_{\lambda_\varepsilon})|$ that depend on the ‘‘approximate sparsity’’ of the true penalized solution λ_ε . Since we are dealing with a loss ℓ of quadratic type, bounding the random error is essentially equivalent to bounding the norm $\|f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}\|_{L_2(\Pi)}$ (see Proposition 1). At the same time, we provide upper bounds on the symmetrized Kullback-Leibler distance between $\hat{\lambda}_\varepsilon$ and λ_ε and show that the ‘‘approximate sparsity’’ properties of these two functions are closely related.

Let \mathcal{H}' be a measurable subset of \mathcal{H} . In the theorem below, it will be a subset of the dictionary \mathcal{H} on which both $\hat{\lambda}_\varepsilon$ and λ_ε are approximately concentrated. Let L be a finite dimensional subspace of $L_2(\Pi)$ that will be used to approximate the functions from \mathcal{H}' . Let $d := \dim(L)$ and denote $U_L(x) := \sup_{h \in L, \|h\|_{L_2(\Pi)} \leq 1} |h(x)|$. It is easy to check (using the Cauchy-Schwarz inequality) that $\|U_L\|_{L_2(\Pi)} = \sqrt{d}$. Denote $U(L) := \|U_L\|_{L_\infty} + 1$. Note that $U(L)$ is of the order \sqrt{d} if there exists an orthonormal basis ϕ_1, \dots, ϕ_d of L such that the functions ϕ_j are uniformly bounded by a constant. Finally, let $\rho(\mathcal{H}'; L) := \sup_{h \in \mathcal{H}'} \|P_{L^\perp} h\|_{L_2(\Pi)}$, where P_{L^\perp} stands for the orthogonal projection on L^\perp . We are interested in those subspaces L for which d and $U(L)$ are not very large and $\rho(\mathcal{H}'; L)$ is small enough, i.e., the space L provides a reasonably good $L_2(\Pi)$ -approximation of the functions from \mathcal{H}' . A natural choice of L might be a subspace spanned on the centers of the $L_2(\Pi)$ -balls of small enough radius δ covering \mathcal{H}' ; in this case $\rho(\mathcal{H}'; L) \leq \delta$ and d is equal to the cardinality of such a δ -covering.

For a function class \mathcal{G} and a probability measure Q on S , let $N(\mathcal{G}; L_2(Q); \varepsilon)$ denote the minimal number of $L_2(Q)$ -balls of radius ε covering \mathcal{G} . We will need the following **complexity assumption** on the base class \mathcal{H} : there exists a nonnegative nonincreasing function Ω such that $\Omega(u) \rightarrow \infty$ as $u \rightarrow 0$, Ω is a regularly varying function of exponent $\alpha \in [0, 2)$ and, with probability 1,

$$\log N(\mathcal{H}; L_2(\Pi_n); u/2) \leq \Omega(u), \quad u > 0, \quad n \in \mathbb{N}. \quad (4.1)$$

In particular, for VC-type classes of VC-dimension V such a bound holds with $\Omega(u)$ of the order $V \log(1/u)$.

Theorem 2 Suppose that the complexity assumption (4.1) holds. There exist constants $C, D > 0$ depending only on ℓ such that for all measurable subsets $\mathcal{H}' \subset \mathcal{H}$, for all finite dimensional subspaces $L \subset L_2(\Pi)$ with $d := \dim(L)$ and $\rho := \rho(\mathcal{H}'; L)$, for all

$$\varepsilon \geq D \sqrt{\frac{\Omega(1/\sqrt{d})}{n}}$$

and for all $t > 0$, the following bounds hold with probability at least $1 - e^{-t}$:

$$\hat{\Lambda}_\varepsilon(\mathcal{H} \setminus \mathcal{H}') \leq C \left[\Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}') \sqrt{\frac{d+t_n}{n\varepsilon}} \sqrt{\frac{\rho}{\varepsilon}} \sqrt{\frac{\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n\varepsilon}} \right], \quad (4.2)$$

$$\Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}') \leq C \left[\hat{\Lambda}_\varepsilon(\mathcal{H} \setminus \mathcal{H}') \sqrt{\frac{d+t_n}{n\varepsilon}} \sqrt{\frac{\rho}{\varepsilon}} \sqrt{\frac{\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n\varepsilon}} \right] \quad (4.3)$$

and

$$\begin{aligned} \|f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}\|_{L_2(\Pi)}^2 + \varepsilon K(\hat{\lambda}_\varepsilon, \lambda_\varepsilon) &\leq C \left[\frac{d+t_n}{n} \sqrt{\frac{\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n\varepsilon}} \right. \\ &\left. \Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}') \sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n}} \right], \end{aligned} \quad (4.4)$$

where $t_n := t + 4 \log \log_2 n + 2 \log 2$.

Theorems 1, 2 and Proposition 1 yield the following **sparsity oracle inequality** for the excess risk of $f_{\hat{\lambda}_\varepsilon}$: for all oracles $\lambda \in \mathbb{D}$, with probability at least $1 - e^{-t}$,

$$\begin{aligned} \mathcal{E}(f_{\hat{\lambda}_\varepsilon}) &\leq 2\mathcal{E}(f_\lambda) + C \left[\frac{d+t_n}{n} \sqrt{\frac{\Omega(\rho/\sqrt{d})}{n}} \sqrt{\Lambda(\mathcal{H} \setminus \mathcal{H}') \sqrt{\frac{\Omega(1/\sqrt{d})}{n}}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n}} \right. \\ &\left. \sqrt{\varepsilon^2 \gamma^2 (\log \lambda)} \right]. \end{aligned}$$

Proof: Let λ_ε be the solution of (1.1) and $\hat{\lambda}_\varepsilon$ be the solution of (1.2). Denote

$$\Lambda_\varepsilon(A) := \int_A \lambda_\varepsilon(h) \mu(dh), \quad \hat{\Lambda}_\varepsilon(A) := \int_A \hat{\lambda}_\varepsilon(h) \mu(dh).$$

Using (2.1) and (2.2), for all $\tau \in (0, 1)$, the directional derivative of F exists at the point $\lambda_\varepsilon + \tau \hat{\lambda}_\varepsilon$ in the direction $\hat{\lambda}_\varepsilon - \lambda_\varepsilon$ and

$$\begin{aligned} DF(\lambda_\varepsilon + \tau(\hat{\lambda}_\varepsilon - \lambda_\varepsilon); \hat{\lambda}_\varepsilon - \lambda_\varepsilon) &= \\ P(\ell' \bullet f_{\lambda_\varepsilon + \tau(\hat{\lambda}_\varepsilon - \lambda_\varepsilon)})(f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}) + \varepsilon \int_{\mathcal{H}} (\hat{\lambda}_\varepsilon - \lambda_\varepsilon) \log(\lambda_\varepsilon + \tau(\hat{\lambda}_\varepsilon - \lambda_\varepsilon)) d\mu &\geq 0. \end{aligned} \quad (4.5)$$

[Note that the directional derivative of entropy H in the direction of $\hat{\lambda}_\varepsilon - \lambda_\varepsilon$ does not necessarily exist at the point λ_ε itself which explains the need in a somewhat more complicated argument given here]. Moreover, since the function $[0, 1] \ni \tau \mapsto F(\lambda_\varepsilon + \tau(\hat{\lambda}_\varepsilon - \lambda_\varepsilon))$ is convex, its right derivative, which coincides with $DF(\lambda_\varepsilon + \tau(\hat{\lambda}_\varepsilon - \lambda_\varepsilon); \hat{\lambda}_\varepsilon - \lambda_\varepsilon)$, is nondecreasing in $\tau \in [0, 1]$. Since λ_ε is the minimal point of F , this implies that, for $\tau \in (0, 1)$,

$$\begin{aligned} DF(\lambda_\varepsilon + \tau(\hat{\lambda}_\varepsilon - \lambda_\varepsilon); \hat{\lambda}_\varepsilon - \lambda_\varepsilon) &= \\ = P(\ell' \bullet f_{\lambda_\varepsilon + \tau(\hat{\lambda}_\varepsilon - \lambda_\varepsilon)})(f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}) + \varepsilon \int_{\mathcal{H}} (\hat{\lambda}_\varepsilon - \lambda_\varepsilon) \log(\lambda_\varepsilon + \tau(\hat{\lambda}_\varepsilon - \lambda_\varepsilon)) d\mu &\geq 0. \end{aligned} \quad (4.6)$$

A similar argument shows that for all $\tau \in (0, 1)$

$$\begin{aligned} DF_n(\hat{\lambda}_\varepsilon + \tau(\lambda_\varepsilon - \hat{\lambda}_\varepsilon); \hat{\lambda}_\varepsilon - \lambda_\varepsilon) &= \\ P_n(\ell' \bullet f_{\hat{\lambda}_\varepsilon + \tau(\lambda_\varepsilon - \hat{\lambda}_\varepsilon)})(f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}) + \varepsilon \int_{\mathcal{H}} (\hat{\lambda}_\varepsilon - \lambda_\varepsilon) \log(\hat{\lambda}_\varepsilon + \tau(\lambda_\varepsilon - \hat{\lambda}_\varepsilon)) d\mu &\leq 0. \end{aligned} \quad (4.7)$$

Subtracting (4.6) from (4.7) and rearranging the terms, we get

$$P\left(\ell' \bullet f_{\hat{\lambda}_\varepsilon + \tau(\lambda_\varepsilon - \hat{\lambda}_\varepsilon)} - \ell' \bullet f_{\lambda_\varepsilon + \tau(\hat{\lambda}_\varepsilon - \lambda_\varepsilon)}\right)(f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}) + \varepsilon \int \left(\hat{\lambda}_\varepsilon - \lambda_\varepsilon\right) \log \frac{(1-\tau)\hat{\lambda}_\varepsilon + \tau\lambda_\varepsilon}{(1-\tau)\lambda_\varepsilon + \tau\hat{\lambda}_\varepsilon} d\mu \leq \left|(P - P_n)(\ell' \bullet f_{\hat{\lambda}_\varepsilon + \tau(\lambda_\varepsilon - \hat{\lambda}_\varepsilon)})(f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon})\right|. \quad (4.8)$$

Under the assumptions on the loss (in particular, continuity of ℓ'), passing to the limit as $\tau \rightarrow 0$, using the dominated convergence, equation (2.3) of Proposition 3 and the bound

$$P\left(\ell' \bullet f_{\hat{\lambda}_\varepsilon} - \ell' \bullet f_{\lambda_\varepsilon}\right)(f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}) \geq c\|f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}\|_{L_2(\Pi)}^2$$

that holds for losses of quadratic type, we get

$$c\|f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}\|_{L_2(\Pi)}^2 + \varepsilon K(\hat{\lambda}_\varepsilon, \lambda_\varepsilon) \leq \left|(P - P_n)(\ell' \bullet f_{\hat{\lambda}_\varepsilon})(f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon})\right|. \quad (4.9)$$

To complete the proof of the theorem, it remains to bound $\left|(P - P_n)(\ell' \bullet f_{\hat{\lambda}_\varepsilon})(f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon})\right|$. Let

$$\Lambda(\delta, \Delta) := \left\{ \lambda \in \mathbb{D} : \|f_\lambda - f_{\lambda_\varepsilon}\|_{L_2(\Pi)} \leq \delta, \int_{\mathcal{H} \setminus \mathcal{H}'} \lambda(h) \mu(dh) \leq \Delta \right\}$$

and

$$\alpha_n(\delta, \Delta) := \sup \{|(P_n - P)(\ell' \bullet f_\lambda)(f_\lambda - f_{\lambda_\varepsilon})|, \lambda \in \Lambda(\delta, \Delta)\}.$$

In what follows we will use Rademacher processes

$$R_n(f) := n^{-1} \sum_{j=1}^n \varepsilon_j f(X_j),$$

where $\{\varepsilon_j\}$ is a sequence of i.i.d. Rademacher random variables (taking values $+1$ and -1 with probability $1/2$) independent of $\{X_j\}$.

Lemma 3 *Let \mathcal{H} be a class of functions on S uniformly bounded by 1 and let $L \subset L_2(\Pi)$ be a finite dimensional subspace with $d := \dim(L)$ and $\rho := \rho(\mathcal{H}; L)$. Suppose that assumption (4.1) holds for some function Ω . Then with some constant $C > 0$*

$$\mathbb{E} \sup_{h \in \mathcal{H}} |R_n(P_{L^\perp} h)| \leq C \left[\rho \sqrt{\frac{\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n}} \right].$$

Proof: The following is true for all $h_1, h_2 \in \mathcal{H}$

$$|P_L(h_1)(x) - P_L(h_2)(x)| \leq U_L(x) \|P_L(h_1) - P_L(h_2)\|_{L_2(\Pi)} \leq U_L(x) \|h_1 - h_2\|_{L_2(\Pi)}$$

and it implies that

$$\|P_L(h_1) - P_L(h_2)\|_{L_2(\Pi_n)} \leq \|U_L\|_{L_2(\Pi_n)} \|h_1 - h_2\|_{L_2(\Pi)}.$$

Therefore,

$$\log N(P_L(\mathcal{H}); L_2(\Pi_n); u) \leq \log N\left(\mathcal{H}; L_2(\Pi); \frac{u}{\|U_L\|_{L_2(\Pi_n)}}\right). \quad (4.10)$$

Complexity assumption (4.1), together with the law of large numbers, gives the bound for covering numbers with respect to $L_2(\Pi)$ (see the proof of Theorem 3.4 in [GK06]):

$$\log N(\mathcal{H}; L_2(\Pi), u) \leq \Omega(u). \quad (4.11)$$

Since $P_{L^\perp} h = h - P_L h$, (4.10) implies

$$N(P_{L^\perp}(\mathcal{H}); L_2(\Pi_n); u) \leq N(\mathcal{H}; L_2(\Pi_n), u/2) N\left(\mathcal{H}; L_2(\Pi); \frac{u}{2\|U_L\|_{L_2(\Pi_n)}}\right).$$

Recalling the complexity conditions (4.1) and (4.11), we easily get

$$\log N(P_{L^\perp}(\mathcal{H}); L_2(\Pi_n); u) \leq \Omega(u) + \Omega\left(\frac{u}{2\|U_L\|_{L_2(\Pi_n)}}\right).$$

It remains to use Theorem 3.1 from [GK06] to complete the proof. \square

Lemma 4 *Under the assumptions of Theorem 2, there exists a constant $C > 0$ depending only on the loss such that with probability at least $1 - e^{-t}$ for all $\frac{1}{\sqrt{n}} \leq \delta \leq 1$, $\frac{1}{\sqrt{n}} \leq \Delta \leq 1$*

$$\alpha_n(\delta, \Delta) \leq C \left[\delta \sqrt{\frac{d+t_n}{n}} \sqrt{\rho} \sqrt{\frac{\Omega(\rho/\sqrt{d})}{n}} \sqrt{\Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}')} \sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\Delta} \sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{t_n}{n}} \right] =: \hat{\beta}_n(\delta, \Delta).$$

where $t_n := t + 4 \log \log_2 n + 2 \log 2$.

Proof: Recall that $\alpha_n(\delta, \Delta) := \sup \{|(P - P_n)(\ell' \bullet f_\lambda)(f_\lambda - f_{\lambda_\varepsilon})|, \lambda \in \Lambda(\delta, \Delta)\}$. The function $u \mapsto \ell'(y, f_{\lambda_\varepsilon} + u)u$, $|u| \leq 2$ is Lipschitz with Lipschitz constant depending only on ℓ . Note that $\ell'(y, f_\lambda(\cdot))(f_\lambda(\cdot) - f_{\lambda_\varepsilon}(\cdot)) = \ell'(y, f_{\lambda_\varepsilon} + u)u|_{u=f_\lambda(\cdot) - f_{\lambda_\varepsilon}(\cdot)}$. This allows us to apply the symmetrization and contraction inequalities (see [vdVW96], Lemma 2.3.6 and Proposition A.3.2) which results in the following bound:

$$\mathbb{E} \alpha_n(\delta, \Delta) \leq C \mathbb{E} \sup_{\lambda \in \Lambda(\delta, \Delta)} |R_n(f_\lambda - f_{\lambda_\varepsilon})|,$$

where $C > 0$ is a constant depending only on ℓ . Let P_L denote the orthogonal projection on a d -dimensional subspace L . The following representation is straightforward:

$$f_\lambda - f_{\lambda_\varepsilon} = P_L(f_\lambda - f_{\lambda_\varepsilon}) + \int_{\mathcal{H}'} P_{L^\perp}(h)(\lambda(h) - \lambda_\varepsilon(h)) \mu(dh) + \int_{\mathcal{H} \setminus \mathcal{H}'} P_{L^\perp}(h)(\lambda(h) - \lambda_\varepsilon(h)) \mu(dh). \quad (4.12)$$

Hence, it is enough to bound separately the expected supremum of the Rademacher process R_n for each term in the sum. For the first term, the standard bound on Rademacher processes indexed by a finite dimensional subspace (see, e.g., [Kol08], proposition 3.2) yields

$$\mathbb{E} \sup_{\lambda \in \Lambda(\delta, \Delta)} |R_n(P_L(f_\lambda - f_{\lambda_\varepsilon}))| \leq \delta \sqrt{\frac{d}{n}}. \quad (4.13)$$

To bound the remaining terms, we will use Lemma 3. First, due to linearity of the Rademacher process,

$$\mathbb{E} \sup_{\lambda \in \Lambda(\delta, \Delta)} \left| R_n \left(\int_{\mathcal{H} \setminus \mathcal{H}'} (\lambda - \lambda_\varepsilon)(h) P_{L^\perp} h \mu(dh) \right) \right| \leq (\Delta + \Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}')) \mathbb{E} \sup_{h \in \mathcal{H} \setminus \mathcal{H}'} |R_n(P_{L^\perp} h)|. \quad (4.14)$$

We now use the bound of Lemma 3 with $\mathcal{H} \setminus \mathcal{H}'$ instead of \mathcal{H} and with $\rho = 1$ to get

$$\begin{aligned} \mathbb{E} \sup_{\lambda \in \Lambda(\delta, \Delta)} \left| R_n \left(\int_{\mathcal{H} \setminus \mathcal{H}'} (\lambda - \lambda_\varepsilon)(h) P_{L^\perp} h \mu(dh) \right) \right| &\leq \\ C(\Delta + \Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}')) &\left[\sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(1/\sqrt{d})}{n}} \right]. \end{aligned} \quad (4.15)$$

Similarly,

$$\mathbb{E} \sup_{\lambda \in \Lambda(\delta, \Delta)} \left| R_n \left(\int_{\mathcal{H}'} (\lambda - \lambda_\varepsilon) P_{L^\perp} h d\mu(h) \right) \right| \leq 2 \mathbb{E} \sup_{h \in \mathcal{H}'} |R_n(P_{L^\perp} h)|$$

and using the bound of Lemma 3 with \mathcal{H}' instead of \mathcal{H} and with $\rho := \rho(\mathcal{H}', L)$, we get

$$\mathbb{E} \sup_{\lambda \in \Lambda(\delta, \Delta)} \left| R_n \left(\int_{\mathcal{H}'} (\lambda - \lambda_\varepsilon)(h) P_{L^\perp} h d\mu(h) \right) \right| \leq C \left[\rho \sqrt{\frac{\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n}} \right]. \quad (4.16)$$

Combining (4.13)–(4.16) results in the following bound:

$$\begin{aligned} \mathbb{E} \alpha_n(\delta, \Delta) \leq C & \left[\delta \sqrt{\frac{d}{n}} \sqrt{\frac{\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n}} \right. \\ & \left. \Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}') \sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\Delta} \sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n}} \right]. \end{aligned} \quad (4.17)$$

Talagrand's concentration inequality (see, e.g., [Bou02]) implies that with probability at least $1 - e^{-s}$ and with a proper choice of constant $C > 0$

$$\alpha_n(\delta, \Delta) \leq \beta_n(\delta, \Delta, s) := 2 \left(\mathbb{E} \alpha_n(\delta, \Delta) + C \delta \sqrt{\frac{s}{n}} + C \frac{s}{n} \right). \quad (4.18)$$

We have to make the bound uniform with respect to $\frac{1}{\sqrt{n}} \leq \delta \leq 1$, $\frac{1}{\sqrt{n}} \leq \Delta \leq 1$. To this end, let

$$\delta_j = \Delta_j = \frac{1}{2^j}, \quad t_{i,j} = t + 2 \log(i+1) + 2 \log(j+1) + 2 \log 2, \quad i, j \geq 0. \quad (4.19)$$

Then, with probability at least

$$1 - \sum_{i,j:\delta_i, \Delta_j \geq n^{-1/2}} \exp\{-t_{i,j}\} \geq 1 - e^{-t - \log^4(\sum_{j \geq 0} (j+1)^{-2})} \geq 1 - e^{-t},$$

for all i, j such that $\delta_i, \Delta_j \geq n^{-1/2}$ and all δ, Δ such that $\delta \in (\delta_{i+1}, \delta_i]$, $\Delta \in (\Delta_{j+1}, \Delta_j]$, the following bounds hold: $\alpha(\delta, \Delta) \leq \beta(\delta_i, \Delta_j, t_{i,j})$. Note that

$$\begin{aligned} t_{i,j} &\leq t + 2 \log 2 + 2 \log \log_2 \left(\frac{1}{\delta} \right) + 2 \log \log_2 \left(\frac{1}{\Delta} \right), \\ \frac{2 \log \log_2 \left(\frac{1}{\Delta} \right)}{n} &\leq 2 \frac{\log \log_2(n)}{n}, \quad \frac{2 \log \log_2 \left(\frac{1}{\delta} \right)}{n} \leq 2 \frac{\log \log_2(n)}{n}, \end{aligned}$$

implying that $t_{i,j} \leq t_n$. Thus, with probability at least $1 - e^{-t}$, for all $\delta, \Delta \in [n^{-1/2}, 1]$

$$\begin{aligned} \alpha_n(\delta, \Delta) \leq \hat{\beta}_n(\delta, \Delta) &:= C \left[\delta \sqrt{\frac{d+t_n}{n}} \sqrt{\frac{\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{t_n}{n}} \right. \\ & \left. \Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}') \sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\Delta} \sqrt{\frac{\Omega(1/\sqrt{d})}{n}} \sqrt{\frac{U(L)\Omega(\rho/\sqrt{d})}{n}} \sqrt{\frac{t_n}{n}} \right]. \end{aligned}$$

□

To complete the proof of the theorem, denote $\hat{\delta} := \|f_{\hat{\lambda}_\varepsilon} - f_{\lambda_\varepsilon}\|_{L_2(\Pi)}$, and $\hat{\Delta} := \hat{\Lambda}_\varepsilon(\mathcal{H} \setminus \mathcal{H}')$. By Lemma 4, (4.9) and (2.4) of Proposition 3, the following inequalities hold with probability at least $1 - e^{-t}$

$$c\hat{\delta}^2 \leq \hat{\beta}_n(\hat{\delta}, \hat{\Delta}), \quad (4.20)$$

$$\varepsilon \hat{\Delta} \leq 2\varepsilon \Lambda_\varepsilon(\mathcal{H} \setminus \mathcal{H}') + \hat{\beta}_n(\hat{\delta}, \hat{\Delta}), \quad (4.21)$$

provided that $\hat{\delta} \geq n^{-1/2}$, $\hat{\Delta} \geq n^{-1/2}$. It remains to solve (4.20), (4.21) for $\hat{\delta}, \hat{\Delta}$ (using the assumption on ε) to get the desired bounds (in the cases when $\hat{\delta} < n^{-1/2}$ and/or $\hat{\Delta} < n^{-1/2}$ the derivation becomes even simpler). □

References

- [Bou02] O. Bousquet. A Bennett concentration inequality and its application to suprema of empirical processes. *C. R. Math. Acad. Sci. Paris*, 334(6):495–500, 2002.
- [BRT09] P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Ann. Statist.*, 37(4):1705–1732, 2009.
- [GK06] E. Giné and V. Koltchinskii. Concentration inequalities and asymptotic results for ratio type empirical processes. *Ann. Probab.*, 34(3):1143–1216, 2006.
- [Kol08] V. Koltchinskii. Oracle inequalities in empirical risk minimization and sparse recovery problems. *Lecture Notes for Ecole d’Eté de Probabilités de Saint-Flour*, 2008.
- [Kol09a] V. Koltchinskii. Sparse recovery in convex hulls via entropy penalization. *Ann. Statist.*, 37(3):1332–1359, 2009.
- [Kol09b] V. Koltchinskii. Sparsity in penalized empirical risk minimization. *Annales Inst. H. Poincaré, Probabilités et Statistique*, 45(1):7–57, 2009.
- [KY08] V. Koltchinskii and M. Yuan. Sparse recovery in large ensembles of kernel machines. In *Proceedings of 19th Annual Conference on Learning Theory (COLT 2008)*, pages 229–238, 2008.
- [MPTJ07] S. Mendelson, A. Pajor, and N. Tomczak-Jaegermann. Reconstruction and subgaussian operators in asymptotic geometric analysis. *Geom. Funct. Anal.*, 17(4):1248–1282, 2007.
- [MvdGB09] L. Meier, S. van de Geer, and P. Bühlmann. High-dimensional additive modeling. *Ann. Statist.*, 37(6B):3779–3821, 2009.
- [RSSZ07] S. Rosset, G. Swirszcz, N. Srebro, and J. Zhu. ℓ_1 regularization in infinite dimensional feature spaces. In *Learning theory*, volume 4539 of *Lecture Notes in Comput. Sci.*, pages 544–558. Springer, Berlin, 2007.
- [vdVW96] A. W. van der Vaart and J. A. Wellner. *Weak convergence and empirical processes*. Springer, 1996.
- [Zha01] T. Zhang. Regularized winnow methods. In *Advances in Neural Information Processing Systems 13 (NIPS 2000)*, pages 703–709. MIT Press, 2001.

Efficient classification for metric data

Lee-Ad Gottlieb*
Weizmann Institute of Science
lee-ad.gottlieb@weizmann.ac.il

Aryeh (Leonid) Kontorovich
Ben Gurion University
karyeh@cs.bgu.ac.il

Robert Krauthgamer*
Weizmann Institute of Science
robert.krauthgamer@weizmann.ac.il

Abstract

Recent advances in large-margin classification of data residing in general metric spaces (rather than Hilbert spaces) enable classification under various natural metrics, such as edit and earthmover distance. The general framework developed for this purpose by von Luxburg and Bousquet [JMLR, 2004] left open the question of computational efficiency and providing direct bounds on classification error.

We design a new algorithm for classification in general metric spaces, whose runtime and accuracy depend on the doubling dimension of the data points. It thus achieves superior classification performance in many common scenarios. The algorithmic core of our approach is an approximate (rather than exact) solution to the classical problems of Lipschitz extension and of Nearest Neighbor Search. The algorithm's generalization performance is established via the fat-shattering dimension of Lipschitz classifiers.

1 Introduction

A recent line of work extends the large-margin classification paradigm from Hilbert spaces to less structured ones, such as Banach or even metric spaces [HBS05, vLB04, DL07]. In this metric approach, data is presented as points with distances but without requiring the additional structure of inner products. The potentially significant advantage is that the metric can be carefully suited to the type of data, e.g. earthmover distance for images, or edit distance for sequences.

However, much of the existing machinery of generalization bounds [CV95, SS02] depends strongly on the inner-product structure of the Hilbert space. von Luxburg and Bousquet [vLB04] developed a powerful framework of large-margin classification for a general metric space \mathcal{X} . First, they show that the natural hypotheses (classifiers) to consider in this context are maximally smooth Lipschitz functions; indeed, they reduce classification (of points in a metric space \mathcal{X}) to finding a Lipschitz function ($f : \mathcal{X} \rightarrow \mathbb{R}$) consistent with the data, which is a classic problem in Analysis, known as Lipschitz extension. Next, they establish error bounds in the form of expected-loss. Finally, the computational problem of evaluating the classification function is reduced, assuming zero training error, to exact 1-nearest neighbor search. This matches a common classification heuristic, see e.g. [CH67], and the analysis of [vLB04] may be viewed as a rigorous explanation for the empirical success of this heuristic.

An important question left open by the work of [vLB04] is the efficient computation of the classifier. Specifically, exact nearest neighbor search in general metrics might require time that is linear in the sample size, and it is algorithmically nontrivial to deal with training error. In particular, the task of choosing which points will be misclassified by the hypothesis (i.e. optimizing the bias-variance tradeoff) remains to be addressed.

Our contribution. We solve the problems delineated above by showing that data with a low doubling dimension admits accurate and computationally efficient classification. In fact, this is the first time in which the doubling dimension of the data points is tied to either classification error or algorithmic runtime. (Previously, the doubling dimension of the space of classifiers was controlled by the VC dimension of the classifier space [BLL09].) We first give an alternate generalization bound for Lipschitz classifiers, which directly bounds the classification error, rather than expected loss. (A similar bound can in fact be derived from the analysis of [vLB04].) Our bound is based on an elementary analysis of the fat-shattering dimension, see Section 3.

*This work was supported in part by The Israel Science Foundation (grant #452/08), and by a Minerva grant.

We then present our main contribution, and give an efficient computational implementation of the Lipschitz classifier. In Section 4 we prove that once a Lipschitz classifier has been chosen, the classifier can be computed (evaluated) quickly on any new point $x \in \mathcal{X}$, by utilizing approximate nearest neighbor search (which is known to be fast when points have a low doubling dimension). In Section 5 we further show how to quickly compute a near-optimal classifier (in terms of classification error bound), even when the training error is nonzero. In particular, this necessitates the optimization of the number of incorrectly labeled examples – and moreover, their identity – as part of the bias-variance tradeoff. In Section 6 we give an example to illustrate the potential power of our approach.

2 Definitions and notation

We use standard notation and definitions throughout.

Metric spaces. A *metric* ρ on a set \mathcal{X} is a positive symmetric function satisfying the triangle inequality $\rho(x, y) \leq \rho(x, z) + \rho(z, y)$; together the two comprise the metric space (\mathcal{X}, ρ) . The diameter of a set $A \subseteq \mathcal{X}$, is defined by $\text{diam}(A) = \sup_{x, y \in A} \rho(x, y)$. The Lipschitz constant of a function $f : \mathcal{X} \rightarrow \mathbb{R}$, denoted by $\|f\|_{\text{Lip}}$, is defined to be the smallest $L > 0$ that satisfies $|f(x) - f(y)| \leq L\rho(x, y)$ for all $x, y \in \mathcal{X}$.

Doubling dimension. For a metric (\mathcal{X}, ρ) , let λ be the smallest value such that every ball in \mathcal{X} can be covered by λ balls of half the radius. The *doubling dimension* of \mathcal{X} is $\text{ddim}(\mathcal{X}) = \log_2 \lambda$. A metric is *doubling* when its doubling dimension is bounded. Note that while a low Euclidean dimension implies a low doubling dimension (Euclidean metrics of dimension d have doubling dimension $O(d)$ [GKL03]), low doubling dimension is strictly more general than low Euclidean dimension.

The following packing property can be demonstrated via a repetitive application of the doubling property: For set S with doubling dimension $\text{ddim}(\mathcal{X})$, if the minimum interpoint distance in S is at least α , and $\text{diam}(S) \leq \beta$, then $|S| \leq \lceil \beta/\alpha \rceil^{\text{ddim}(\mathcal{X})+1}$ (see, for example [KL04]).

Learning. Our setting in this paper is a generalization of PAC known as *probabilistic concept learning* [KS94]. In this model, examples are drawn independently from $\mathcal{X} \times \{-1, 1\}$ according to some unknown probability distribution P , and the learner, having observed n such pairs (x, y) produces a hypothesis $h : \mathcal{X} \rightarrow \{-1, 1\}$. The *generalization error* is the probability of misclassifying a new point drawn from P :

$$P \{(x, y) : h(x) \neq y\}.$$

The quantity above is random (since it depends on a random sequence) and we wish to upper-bound it in probability. Most bounds of this sort contains a *sample error* term (corresponding in statistics to bias), which is the fraction of observed examples misclassified by h and a *hypothesis complexity* term (corresponding to variance in statistics) which measures the richness of the class of all admissible hypotheses [Was06]. Keeping in line with the literature, we ignore the measure-theoretic technicalities associated with taking suprema over uncountable function classes.

3 Generalization bounds

In this section, we take a preliminary step towards our efficient classification algorithm by deriving generalization bounds for Lipschitz classifiers over doubling spaces. As noted by [vLB04] Lipschitz functions are the natural object to consider in an optimization/regularization framework. The basic intuition behind our proofs is that the Lipschitz constant plays the role of the inverse margin in the confidence of the classifier. As in [vLB04], small Lipschitz constant corresponds to large margin, which in turn yields low hypothesis complexity and variance. In retrospect, our generalization bound (Corollary 5 below) can be derived as a consequence of [vLB04, Theorem 18] in conjunction with [BM02, Theorem 5(b)].

We apply tools from generalized Vapnik-Chervonenkis theory to the case of Lipschitz classifiers. Let \mathcal{F} be a collection of functions $f : \mathcal{X} \rightarrow \mathbb{R}$ and recall the definition of the fat-shattering dimension [ABCH97, BS99]: a set $X \subset \mathcal{X}$ is said to be γ -shattered by \mathcal{F} if there exists some function $r : X \rightarrow \mathbb{R}$ such that for each label assignment $y \in \{-1, 1\}^X$ there is an $f \in \mathcal{F}$ satisfying $y(x)(f(x) - r(x)) \geq \gamma > 0$ for all $x \in X$. The γ -fat-shattering dimension of \mathcal{F} , denoted by $\text{fat}_\gamma(\mathcal{F})$, is the cardinality of the largest set γ -shattered by \mathcal{F} .

For the case of Lipschitz functions, we will show that the notion of fat-shattering dimension may be somewhat simplified. We say that a set $X \subset \mathcal{X}$ is γ -shattered *at zero* by a collection of functions \mathcal{F} if for each $y \in \{-1, 1\}^X$ there is an $f \in \mathcal{F}$ satisfying $y(x)f(x) \geq \gamma$ for all $x \in X$. (This is the definition above with $r \equiv 0$.) We write $\text{fat}_\gamma^0(\mathcal{F})$ to denote the cardinality of the largest set γ -shattered at zero by \mathcal{F} and show that for Lipschitz function classes the two complexity measures are the same.

Lemma 1 *Let \mathcal{F} be the collection of all $f : \mathcal{X} \rightarrow \mathbb{R}$ with $\|f\|_{\text{Lip}} \leq L$. Then $\text{fat}_\gamma(\mathcal{F}) = \text{fat}_\gamma^0(\mathcal{F})$.*

Proof: We begin by recalling the classic Lipschitz extension result, essentially due to McShane and Whitney [McS34, Whi34]. Any real-valued function f defined on a subset X of a metric space \mathcal{X} has an extension f^* to all of \mathcal{X} satisfying $\|f^*\|_{\text{Lip}} = \|f\|_{\text{Lip}}$. Thus, in what follows we will assume that any function f defined on $X \subset \mathcal{X}$ is also defined on all of \mathcal{X} via some Lipschitz extension (in particular, to bound $\|f\|_{\text{Lip}}$ it suffices to bound the restricted $\|f|_X\|_{\text{Lip}}$).

Consider some finite $X \subset \mathcal{X}$. If X is γ -shattered at zero by \mathcal{F} then by definition it is also γ -shattered. Now assume that X is γ -shattered by \mathcal{F} . Thus, there is some function $r : X \rightarrow \mathbb{R}$ such that for each $y \in \{-1, 1\}^X$ there is an $f = f_{r,y} \in \mathcal{F}$ such that $f_{r,y}(x) \geq r(x) + \gamma$ if $y(x) = +1$ and $f_{r,y}(x) \leq r(x) - \gamma$ if $y(x) = -1$. Let us define the function \tilde{f}_y on X and as per above, on all of \mathcal{X} , by $\tilde{f}_y(x) = \gamma y(x)$. It is clear that the collection $\{\tilde{f}_y : y \in \{-1, 1\}^X\}$ γ -fat-shatters X at zero; it only remains to verify that $\tilde{f}_y \in \mathcal{F}$, i.e.,

$$\sup_{y \in \{-1, 1\}^X} \|\tilde{f}_y\|_{\text{Lip}} \leq \sup_{y \in \{-1, 1\}^X} \|f_{r,y}\|_{\text{Lip}}.$$

Indeed,

$$\sup_{y \in \{-1, 1\}^X} \sup_{x, x' \in X} \frac{f_{r,y}(x) - f_{r,y}(x')}{\rho(x, x')} \geq \sup_{x, x' \in X} \frac{r(x) - r(x') + 2\gamma}{\rho(x, x')} \geq \sup_{x, x' \in X} \frac{2\gamma}{\rho(x, x')} = \sup_{y \in \{-1, 1\}^X} \|\tilde{f}_y\|_{\text{Lip}}. \quad \blacksquare$$

A consequence of Lemma 1 is that in considering the generalization properties of Lipschitz functions we need only bound the γ -fat-shattering dimension at zero. The latter follows from the observation that the packing number of a metric space controls the fat-shattering dimension of Lipschitz functions defined over the metric space. Let $M(\mathcal{X}, \rho, \varepsilon)$ be defined as the ε -packing number of \mathcal{X} , the cardinality of the largest ε -separated subset of \mathcal{X} .

Theorem 2 *Let (\mathcal{X}, ρ) be a metric space. Fix some $L > 0$, and let \mathcal{F} be the collection of all $f : \mathcal{X} \rightarrow \mathbb{R}$ with $\|f\|_{\text{Lip}} \leq L$. Then for all $\gamma > 0$,*

$$\text{fat}_\gamma(\mathcal{F}) = \text{fat}_\gamma^0(\mathcal{F}) \leq M(\mathcal{X}, \rho, 2\gamma/L).$$

Proof: Suppose that $S \subseteq \mathcal{X}$ is fat γ -shattered at zero. The case $|S| = 1$ is trivial, so we assume the existence of $x \neq x' \in S$ and $f \in \mathcal{F}$ such that $f(x) \geq \gamma > -\gamma \geq f(x')$. The Lipschitz property then implies that $\rho(x, x') \geq 2\gamma/L$, and the claim follows. \blacksquare

Corollary 3 *Let metric space \mathcal{X} have doubling dimension $\text{ddim}(\mathcal{X})$, and let \mathcal{F} be the collection of real-valued functions over \mathcal{X} with Lipschitz constant at most L . Then for all $\gamma > 0$,*

$$\text{fat}_\gamma(\mathcal{F}) \leq \left\lceil \frac{L \text{diam}(\mathcal{X})}{2\gamma} \right\rceil^{\text{ddim}(\mathcal{X})+1}.$$

Proof: The claim follows immediately from Theorem 2 and the packing property of doubling spaces. \blacksquare

Equipped with these estimates for the fat-shattering dimension of Lipschitz classifiers, we can invoke a standard generalization bound stated in terms of this quantity. For the remainder of this section, we take $\gamma = 1$ and say that a function f classifies an example (x_i, y_i) correctly if

$$y_i f(x_i) \geq 1. \quad (1)$$

The following generalization bounds appear in [BS99]:

Theorem 4 *Let \mathcal{F} be a collection of real-valued functions over some set \mathcal{X} , define $d = \text{fat}_{1/16}(\mathcal{F})$ and let P be some probability distribution on $\mathcal{X} \times \{-1, 1\}$. Suppose that (x_i, y_i) , $i = 1, \dots, n$ are drawn from $\mathcal{X} \times \{-1, 1\}$ independently according to P and that some $f \in \mathcal{F}$ classifies the n examples correctly, in the sense of (1). Then with probability at least $1 - \delta$*

$$P \{(x, y) : \text{sgn}(f(x)) \neq y\} \leq \frac{2}{n} (d \log_2(34en/d) \log_2(578n) + \log_2(4/\delta)).$$

Furthermore, if $f \in \mathcal{F}$ is correct on all but k examples, we have with probability at least $1 - \delta$

$$P \{(x, y) : \text{sgn}(f(x)) \neq y\} \leq k/n + \sqrt{\frac{2}{n} (d \ln(34en/d) \log_2(578n) + \ln(4/\delta))}.$$

Applying Corollary 3, we obtain the following consequence of Theorem 4:

Corollary 5 *Let metric space \mathcal{X} have doubling dimension $\text{ddim}(\mathcal{X})$, and let \mathcal{F} be the collection of real-valued functions over \mathcal{X} with Lipschitz constant at most L . Then for any $f \in \mathcal{F}$ that classifies a sample of size n correctly, we have with probability at least $1 - \delta$*

$$P\{(x, y) : \text{sgn}(f(x)) \neq y\} \leq \frac{2}{n} (d \log_2(34en/d) \log_2(578n) + \log_2(4/\delta)).$$

Likewise, if f is correct on all but k examples, we have with probability at least $1 - \delta$

$$P\{(x, y) : \text{sgn}(f(x)) \neq y\} \leq k/n + \sqrt{\frac{2}{n} (d \ln(34en/d) \log_2(578n) + \ln(4/\delta))}. \quad (2)$$

In both cases, $d = \text{fat}_{1/16}(\mathcal{F}) \leq \lceil 8L \text{diam}(\mathcal{X}) \rceil^{\text{ddim}(\mathcal{X})+1}$.

4 Lipschitz extension classifier

Given a labeled set $(X, Y) \subset \mathcal{X} \times \{-1, 1\}$, we construct our classifier in a similar manner to [vLB04, Lemma 12], via a Lipschitz extension of the labels Y to all of \mathcal{X} . Let $S^+, S^- \subset X$ be the sets of positive and negative labeled points that the classifier correctly labels. Our starting point is the same extension function used in [vLB04], namely, for all $\alpha \in [0, 1]$

$$f_\alpha = \alpha \min_i \left(y_i + 2 \frac{d(x, x_i)}{d(S^+, S^-)} \right) + (1 - \alpha) \max_j \left(y_j - 2 \frac{d(x, x_j)}{d(S^+, S^-)} \right).$$

However, evaluating the exact value of $f_\alpha(x)$ for each point $x \in \mathcal{X}$ (or even the sign of $f_\alpha(x)$ at this point) requires an exact nearest neighbor search, and in arbitrary metric space nearest neighbor search may require $\Theta(|X|)$ time.

In this section, we give a classifier whose sign can be evaluated using a $(1 + \varepsilon)$ -approximate nearest neighbor search. There exists a search structure for an n point set that can be built in $2^{O(\text{ddim}(\mathcal{X}))} n \log n$ time and supports approximate nearest neighbor searches in time $2^{O(\text{ddim}(\mathcal{X}))} \log n + \varepsilon^{-O(\text{ddim}(\mathcal{X}))}$ [CG06, HM06] (see also [KL04, BKL06]). In constructing the classifier, we assume that the sample points have already been partitioned in a manner that yields a favorable bias-variance tradeoff, as in Section 5 below. Therefore, the algorithm below takes as input a set of point $S_1 \subset X$ that must be correctly classified, and a set of error points $S_0 = X - S_1$ that may be ignored in the classifier construction (but which affect the resulting generalization bound).

Theorem 6 *Let \mathcal{X} be a metric space, and fix $0 < \varepsilon \leq \frac{1}{2}$. Given a labeled sample $S = (x_i, y_i) \in \mathcal{X} \times \{-1, 1\}$, $i = 1, \dots, n$, let S be partitioned into S_0 and S_1 , of sizes k and $n - k$, where S_0 contains points that may be misclassified, and S_1 contains points that may not be misclassified. Define $S_1^+, S_1^- \subset S_1$ according to their labels and define $L = 2/d(S_1^+, S_1^-)$. Then there exists a binary classification function $h : \mathcal{X} \rightarrow \{-1, 1\}$ satisfying the following:*

- (a) $h(x)$ can be evaluated at each $x \in \mathcal{X}$ via a single $(1 + \varepsilon)$ -nearest neighbor query. In particular, $h(x)$ can be evaluated in time $2^{O(\text{ddim}(\mathcal{X}))} \log n + \varepsilon^{-O(\text{ddim}(\mathcal{X}))}$, after an initial computation of $(2^{O(\text{ddim}(\mathcal{X}))} \log n + \varepsilon^{-O(\text{ddim}(\mathcal{X}))})n$ time.
- (b) With probability at least $1 - \delta$

$$P\{(x, y) : h(x) \neq y\} \leq 2 \left(\frac{k}{n} + \sqrt{\frac{2}{n} (d \ln(34en/d) \log_2(578n) + \ln(4/\delta))} \right)$$

where $d = \lceil 8(1 + \varepsilon)L \text{diam}(\mathcal{X}) \rceil^{\text{ddim}(\mathcal{X})+1}$.

Proof: Let the distance function $\tilde{d}(\cdot, \cdot)$ be the approximate distance between a point and a set (or between two sets), as determined by a fixed $(1 + \frac{\varepsilon}{4})$ -nearest neighbor search structure. Let

$$\tilde{f}_1(x) := \min_i \left(y_i + 2 \frac{\tilde{d}(x, x_i)}{\tilde{d}(S_1^+, S_1^-)} \right),$$

and let the classifier be $h(x) := \text{sgn}(\tilde{f}_1(x))$. $h(x)$ can be evaluated via an approximate nearest neighbor query in time $2^{O(\text{ddim}(\mathcal{X}))} \log n + \varepsilon^{-O(\text{ddim}(\mathcal{X}))}$, assuming that a search structure has been precomputed in

time $2^{O(\text{ddim}(\mathcal{X}))} n \log n$, and $\tilde{d}(S_1^+, S_1^-)$ has been precomputed via $O(n)$ nearest neighbor searches in time $(2^{O(\text{ddim}(\mathcal{X}))} \log n + \varepsilon^{-O(\text{ddim}(\mathcal{X}))})n$.

It remains to bound the generalization error of h . To this end, define

$$f_1^+(x) = (1 + \varepsilon)f_1(x) + \varepsilon = (1 + \varepsilon) \min_i \left(y_i + 2 \frac{d(x, x_i)}{d(S_1^+, S_1^-)} \right) + \varepsilon,$$

$$f_1^-(x) = (1 + \varepsilon)f_1(x) - \varepsilon = (1 + \varepsilon) \min_i \left(y_i + 2 \frac{d(x, x_i)}{d(S_1^+, S_1^-)} \right) - \varepsilon.$$

Note that $f_1^+(x) > f_1^-(x)$. Both $f_1^+(x)$ and $f_1^-(x)$ correctly classify all labeled points of S_1 and have Lipschitz constant $(1 + \varepsilon)L$, so their classification bounds are given by Corollary 5 with this Lipschitz constant.

We claim that $h(x)$ always agrees with the sign of at least one of $f_1^+(x)$ and $f_1^-(x)$: If $f_1^+(x)$ and $f_1^-(x)$ disagree in their sign, then the claim follows trivially. Assume then that the signs of $f_1^+(x)$ and $f_1^-(x)$ agree. Suppose that $f_1^+(x)$ and $f_1^-(x)$ are positive, which implies that $y_j + 2 \frac{d(x, x_j)}{d(S_1^+, S_1^-)} > \frac{\varepsilon}{1 + \varepsilon}$ for all j . Now recall that $\tilde{f}_1(x) = \min_i \left(y_i + 2 \frac{\tilde{d}(x, x_i)}{d(S^+, S^-)} \right) \geq \min_i \left(y_i + \frac{2}{(1 + \varepsilon/4)^2} \frac{d(x, x_i)}{d(S^+, S^-)} \right)$. If $y_i = +1$, then trivially $h(x)$ is positive. If $y_i = -1$, we have that $2 \frac{d(x, x_i)}{d(S^+, S^-)} > \frac{\varepsilon}{1 + \varepsilon} + 1 = \frac{1 + 2\varepsilon}{1 + \varepsilon}$, and so $\tilde{f}_1(x) \geq \min_i \left(y_i + \frac{2}{(1 + \varepsilon/4)^2} \frac{d(x, x_i)}{d(S^+, S^-)} \right) > -1 + \frac{1}{(1 + \varepsilon/4)^2} \left(\frac{1 + 2\varepsilon}{1 + \varepsilon} \right) > 0$, and we are done. Suppose then that $f_1^+(x)$ and $f_1^-(x)$ are negative, which implies that $y_j + 2 \frac{d(x, x_j)}{d(S_1^+, S_1^-)} < -\frac{\varepsilon}{1 + \varepsilon}$ for some fixed j . Now it must be that $y_j = -1$, and so $2 \frac{d(x, x_j)}{d(S^+, S^-)} < -\frac{\varepsilon}{1 + \varepsilon} + 1 = \frac{1}{1 + \varepsilon}$. Now recall that $\tilde{f}_1(x) = \min_i \left(y_i + 2 \frac{\tilde{d}(x, x_i)}{d(S^+, S^-)} \right) \leq \left(y_j + 2(1 + \varepsilon/4)^2 \frac{d(x, x_j)}{d(S^+, S^-)} \right) < -1 + (1 + \varepsilon/4)^2 \left(\frac{1}{1 + \varepsilon} \right) < 0$, and we are done.

It follows that if $h(x)$ misclassifies x , then x must be misclassified by at least one of $f_1^+(x)$ and $f_1^-(x)$. Hence, the generalization bound of $h(x)$ is not greater than the sum of the generalization bounds of $f_1^+(x)$ and $f_1^-(x)$. \blacksquare

5 Bias-variance tradeoffs

In this section, we show how to efficiently construct a classifier that optimizes the bias-variance tradeoff implicit in Corollary 5, equation (2). Let \mathcal{X} be a metric space, and assume we are given a labeled sample $S = (x_i, y_i) \in \mathcal{X} \times \{-1, 1\}$. For any Lipschitz constant L , let $k(L)$ be the minimal sample error of S over all classifiers with Lipschitz constant L . We rewrite the generalization bound as follows:

$$G(L) = P \{ (x, y) : \text{sgn}(f(x)) \neq y \} \leq k(L)/n + \sqrt{\frac{2}{n} (d \ln(34en/d) \log_2(578n) + \ln(4/\delta))}$$

where $d = \lceil 8L \text{diam}(\mathcal{X}) \rceil^{\text{ddim}(\mathcal{X})+1}$. This bound contains a free parameter, L , which may be tuned to optimize the bias-variance tradeoff. More precisely, decreasing L drives the bias term (number of mistakes) up and the variance term (fat-shattering dimension) down. For some optimal values of L , $G(L)$ achieves a minimum value. The following theorem gives our bias-variance tradeoff.

Theorem 7 *Let \mathcal{X} be a metric space. Given a labeled sample $S = (x_i, y_i) \in \mathcal{X} \times \{-1, 1\}$, $i = 1, \dots, n$, there exists a binary classification function $h : \mathcal{X} \rightarrow \{-1, 1\}$ satisfying the following properties:*

- (a) $h(x)$ can be evaluated at each $x \in \mathcal{X}$ in time $2^{O(\text{ddim}(\mathcal{X}))} \log n$, after an initial computation of $O(n^2 \log n)$ time.
- (b) The generalization error of h is bound by

$$P \{ (x, y) : \text{sgn}(f(x)) \neq y \} \leq c \cdot \inf_{L > 0} \left(k(L)/n + \sqrt{\frac{2}{n} (d \ln(34en/d) \log_2(578n) + \ln(4/\delta))} \right).$$

for some constant c , and where $d = d(L) = \lceil 8L \text{diam}(\mathcal{X}) \rceil^{\text{ddim}(\mathcal{X})+1}$.

We proceed with a description of the algorithm. We will first give an algorithm with runtime $O(n^{4.376})$, and then improve the runtime to $O(n^2 \log n)$.

Algorithm description. Here we give a randomized algorithm that finds an optimal value L^* , that is $G(L^*) = \inf_{L>0} G(L)$. The runtime of this algorithm is $O(n^{4.376})$ with high probability.

First note the behavior of $k(L)$ as L increases. $k(L)$ may decrease when the value of L crosses some critical value: This critical value is determined by point pairs $x_i \in S^+, x_j \in S^-$ (that is, $L = \frac{2}{d(x_i, x_j)}$) and implies that the classification function can correctly classify both these points. There are $O(n^2)$ critical values of L , and these can be determined by enumerating all interpoint distances between sets $S^+, S^- \subset S$.

Below, we will show that for any given L , the value $k(L)$ can be computed in randomized time $O(n^{2.376})$. More precisely, we will show how to compute a partition of S into sets S_1 (with Lipschitz constant L) and S_0 (of size $k(L)$) in this time. Given sets $S_0, S_1 \subset S$, we can construct the classifier of Corollary 5. Since there are $O(n^2)$ critical values of L , we can calculate $k(L)$ for each critical value in $O(n^{4.376})$ total time, and thereby determine L^* . Then by Corollary 5, we may compute a classifier with a bias-variance tradeoff arbitrarily close to optimal.

It is left to describe how value $k(L)$ is computed for any L in randomized time $O(n^{2.376})$. Consider the following algorithm: Construct a bipartite graph $G = (V^+, V^-, E)$. The vertex sets V^+, V^- correspond to the labeled sets $S^+, S^- \in S$, respectively. The length of edge $e = (u, v)$ connecting vertices $u \in V^+$ and $v \in V^-$ is equal to the distance between the points, and E includes all edges of length less than $2/L$. (E can be computed in $O(n^2 \log n)$ time.) Now, for all edges $e \in E$, a classifier with Lipschitz constant L necessarily misclassifies at least one endpoint of e . Hence, the problem of finding a classifier with Lipschitz constant L that misclassifies a minimum number of points in S is equivalent to finding a minimum vertex cover for bipartite graph G . By König's theorem, minimum bipartite vertex cover is itself equivalent to the maximum matching problem on bipartite graphs. An exact solution to the bipartite matching problem may be computed in randomized time $O(n^{2.376})$ [MS04]. This solution immediately identifies sets S_0, S_1 , which allows us to compute a classifier with a bias-variance tradeoff arbitrarily close to optimal.

Improved algorithmic runtime. The runtime given above can be reduced from randomized $O(n^{4.376})$ to deterministic $O(n^2 \log n)$, if we are willing to settle for a generalization bound $G(L)$ within a constant factor of the optimal $G(L^*)$.

The first improvement is in the runtime of the vertex cover algorithm. It is well known that a 2-approximation to the minimum vertex cover on an arbitrary graph can be computed by a greedy algorithm in time $O(|V^+ + V^-| + |E|) = O(n^2)$ [GJ77]. Hence, we may evaluate in $O(n^2)$ time a function $k'(L)$ which satisfies $k(L) \leq k'(L) \leq 2k(L)$.

The second improvement uses a binary search over the values of L , which allows us to evaluate $k'(L)$ for only $O(\log n)$ values of L , as opposed to all $\Theta(n^2)$ values above. Now, we seek the a value of L for which

$$G'(L) = k'(L)/n + \sqrt{\frac{2}{n} (d \ln(34en/d) \log_2(578n) + \ln(4/\delta))}$$

is minimal. Call this value L' . Also note that for all L , $G'(L) \leq 2G(L)$, from which it follows that $G'(L') \leq 2G(L^*)$. While we cannot efficiently find L' , we are able to use a binary search to find a value L for which $G'(L) \leq 2G'(L') \leq 4G(L^*)$. In particular we seek the minimum value of L for which

$$k'(L)/n \leq \sqrt{\frac{2}{n} (d \ln(34en/d) \log_2(578n) + \ln(4/\delta))}.$$

Now, decreasing L can only increase $k'(L)$, so the solution to the inequality above necessarily yields an L for which $G'(L) \leq 2G'(L') \leq 4G(L^*)$. The solution to the inequality can be computed through a binary search on all values of L . By Corollary 5, we can construct a classifier with a bias-variance tradeoff within a factor $4(1 + \varepsilon)$ of optimal. The total runtime is $O(n^2 \log n)$.

6 Example: Earthmover metric

To illustrate the potential power of our approach, we now analyze the doubling dimension of an earthmover metric \mathcal{X}_k that is often used in computer vision applications. ($k \geq 2$ is a parameter.) Each point in \mathcal{X}_k is a multiset of size k in the unit square in the Euclidean plane, formally $S \subset [0, 1]^2$ and $|S| = k$ (allowing and counting multiplicities). The distance between such sets S, T (i.e. two points in \mathcal{X}_k) is given by

$$\text{EMD}(S, T) = \min_{\pi: S \rightarrow T} \left\{ \frac{1}{k} \sum_{s \in S} \|s - \pi(s)\|_2 \right\},$$

where the minimum is over all one-to-one mappings $\pi : S \rightarrow T$. In other words, $\text{EMD}(S, T)$ is the minimum-cost matching between the two sets S, T , where costs correspond to Euclidean distance.

Lemma 8 *The earthmover metric X above satisfies $\text{diam}(\mathcal{X}_k) \leq \sqrt{2}$, and $\text{ddim}(\mathcal{X}_k) \leq O(k \log k)$.*

Proof: For the rest of this proof, a point refers to the unit square, not \mathcal{X}_k . Fix $r > 0$ and consider a ball (in \mathcal{X}_k) of radius r around some S . Let N be an $r/2$ -net of the unit square $[0, 1]^2$. Now consider all multisets T of size k of the unit square which satisfy the following condition: every point in T belongs to the net N and is within (Euclidean) distance $(k + 1/2)r$ from at least one point of S . Points in such a multiset T are chosen from a collection of size at most $k \cdot \left\lceil \frac{(k+1/2)r}{r/2} \right\rceil^{O(1)} \leq k^{O(1)}$ (by the packing property in the Euclidean plane). Thus, the number of such multisets T is at most $\lambda \leq (k^{O(1)})^k = k^{O(k)}$.

We complete the proof of the lemma, by showing that the r -ball (in \mathcal{X}_k) around S is covered by the λ balls of radius $r/2$ whose centers are given by the above multisets T . To see this, consider a multiset S' such that $\text{EMD}(S, S') \leq r$, and let us show that S' is contained in an $r/2$ -ball around one of the above multisets T . Observe that every point in S' is within distance at most kr from at least one point of S . By “mapping” each point in S' to its nearest point in the net N , we get a multiset T as above with $\text{EMD}(S', T) \leq r/2$. ■

References

- [ABCH97] Noga Alon, Shai Ben-David, Nicolò Cesa-Bianchi, and David Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM*, 44(4):615–631, 1997.
- [BKL06] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 97–104, New York, NY, USA, 2006. ACM.
- [BLL09] Nader H. Bshouty, Yi Li, and Philip M. Long. Using the doubling dimension to analyze the generalization of learning algorithms. *Journal of Computer and System Sciences*, 75(6):323–335, 2009.
- [BM02] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.
- [BS99] Peter Bartlett and John Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In *Advances in kernel methods: support vector learning*, pages 43–54, Cambridge, MA, USA, 1999. MIT Press.
- [CG06] Richard Cole and Lee-Ad Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *STOC*, pages 574–583, 2006.
- [CH67] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.
- [CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [DL07] Ricky Der and Daniel Lee. Large-Margin Classification in Banach Spaces. In *JMLR Workshop and Conference Proceedings Volume 2: AISTATS 2007*, pages 91–98, 2007.
- [GJ77] M. R. Garey and D. S. Johnson. The rectilinear steiner tree problem is np -complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.
- [GKL03] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *FOCS*, pages 534–543, 2003.
- [HBS05] Matthias Hein, Olivier Bousquet, and Bernhard Schölkopf. Maximal margin classification for metric spaces. *J. Comput. Syst. Sci.*, 71(3):333–359, 2005.
- [HM06] Sarel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006.
- [KL04] Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
- [KS94] Michael J. Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts. *J. Comput. Syst. Sci.*, 48(3):464–497, 1994.
- [McS34] E. J. McShane. Extension of range of functions. *Bull. Amer. Math. Soc.*, 40(12):837–842, 1934.
- [MS04] Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255, Washington, DC, USA, 2004. IEEE Computer Society.
- [SS02] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2002.
- [vLB04] Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5:669–695, 2004.
- [Was06] Larry Wasserman. *All of nonparametric statistics*. Springer Texts in Statistics. Springer, New York, 2006.

- [Whi34] Hassler Whitney. Analytic extensions of differentiable functions defined in closed sets. *Transactions of the American Mathematical Society*, 36(1):63–89, 1934.

Learning Kernel-Based Halfspaces with the Zero-One Loss

Shai Shalev-Shwartz
The Hebrew University
shais@cs.huji.ac.il

Ohad Shamir
The Hebrew University
ohadsh@cs.huji.ac.il

Karthik Sridharan
Toyota Technological Institute
karthik@tti-c.org

Abstract

We describe and analyze a new algorithm for agnostically learning kernel-based halfspaces with respect to the *zero-one* loss function. Unlike most previous formulations which rely on surrogate convex loss functions (e.g. hinge-loss in SVM and log-loss in logistic regression), we provide finite time/sample guarantees with respect to the more natural zero-one loss function. The proposed algorithm can learn kernel-based halfspaces in worst-case time $\text{poly}(\exp(L \log(L/\epsilon)))$, for *any* distribution, where L is a Lipschitz constant (which can be thought of as the reciprocal of the margin), and the learned classifier is worse than the optimal halfspace by at most ϵ . We also prove a hardness result, showing that under a certain cryptographic assumption, no algorithm can learn kernel-based halfspaces in time polynomial in L .

1 Introduction

A highly important hypothesis class in machine learning theory and applications is that of halfspaces in a Reproducing Kernel Hilbert Space (RKHS). Choosing a halfspace based on empirical data is often performed using Support Vector Machines (SVMs) [25]. SVMs replace the more natural 0-1 loss function with a convex surrogate – the hinge-loss. By doing so, we can rely on convex optimization tools. However, there are no guarantees on how well the hinge-loss approximates the 0-1 loss function. There do exist some recent results on the *asymptotic* relationship between surrogate convex loss functions and the 0-1 loss function [27, 4], but these do not come with finite-sample or finite-time guarantees. In this paper, we tackle the task of learning kernel-based halfspaces with respect to the non-convex 0-1 loss function. Our goal is to derive learning algorithms and to analyze them in the finite-sample finite-time setting.

Following the standard statistical learning framework, we assume that there is an unknown distribution, \mathcal{D} , over the set of labeled examples, $\mathcal{X} \times \{0, 1\}$, and our primary goal is to find a classifier, $h : \mathcal{X} \rightarrow \{0, 1\}$, with low generalization error,

$$\text{err}_{\mathcal{D}}(h) \stackrel{\text{def}}{=} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [|h(\mathbf{x}) - y|] . \quad (1)$$

The learning algorithm is allowed to sample a training set of labeled examples, $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$, where each example is sampled i.i.d. from \mathcal{D} , and it returns a classifier. Following the agnostic PAC learning framework [15], we say that an algorithm (ϵ, δ) -learns a concept class H of classifiers using m examples, if with probability of at least $1 - \delta$ over a random choice of m examples the algorithm returns a classifier \hat{h} that satisfies

$$\text{err}_{\mathcal{D}}(\hat{h}) \leq \inf_{h \in H} \text{err}_{\mathcal{D}}(h) + \epsilon . \quad (2)$$

We note that \hat{h} does not necessarily belong to H . Namely, we are concerned with *improper* learning, which is as useful as proper learning for the purpose of deriving good classifiers. A common learning paradigm is the Empirical Risk Minimization (ERM) rule, which returns a classifier that minimizes the average error over the training set,

$$\hat{h} \in \underset{h \in H}{\text{argmin}} \frac{1}{m} \sum_{i=1}^m |h(\mathbf{x}_i) - y_i| .$$

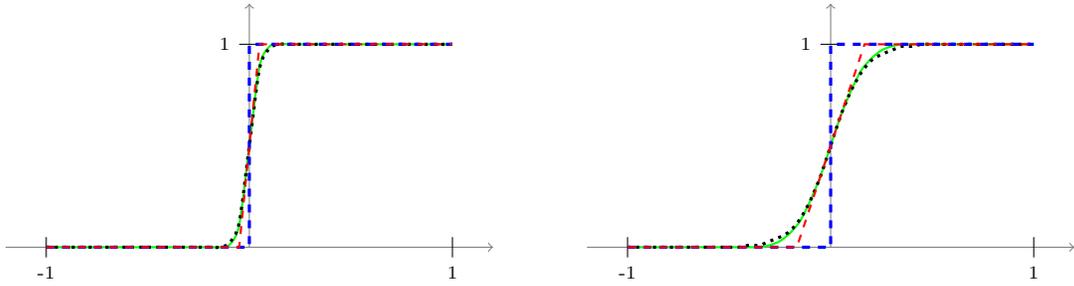


Figure 1: Illustrations of transfer functions for $L = 10$ (left) and $L = 3$ (right): the 0-1 transfer function (dashed blue line); the sigmoid transfer function (dotted black line); the erf transfer function (green line); the piece-wise linear transfer function (dashed red line).

The class of (origin centered) halfspaces is defined as follows. Let \mathcal{X} be a compact subset of a RKHS, which w.l.o.g. will be taken to be the unit ball around the origin. Let $\phi_{0-1} : \mathbb{R} \rightarrow \mathbb{R}$ be the function $\phi_{0-1}(a) = \mathbf{1}(a \geq 0) = \frac{1}{2}(\text{sgn}(a) + 1)$. The class of halfspaces is the set of classifiers

$$H_{\phi_{0-1}} \stackrel{\text{def}}{=} \{ \mathbf{x} \mapsto \phi_{0-1}(\langle \mathbf{w}, \mathbf{x} \rangle) : \mathbf{w} \in \mathcal{X} \} .$$

Although we represent the halfspace using $\mathbf{w} \in \mathcal{X}$, which is a vector in the RKHS whose dimensionality can be infinite, in practice we only need a function that implements inner products in the RKHS (a.k.a. a kernel function), and one can define \mathbf{w} as the coefficients of a linear combination of examples in our training set. To simplify the notation throughout the paper, we represent \mathbf{w} simply as a vector in the RKHS.

It is well known that if the dimensionality of \mathcal{X} is n , then the VC dimension of $H_{\phi_{0-1}}$ equals n . This implies that the number of training examples required to obtain a guarantee of the form given in Equation (2) for the class of halfspaces scales at least linearly with the dimension n [25]. Since kernel-based learning algorithms allow \mathcal{X} to be an infinite dimensional inner product space, we must use a different class in order to obtain a guarantee of the form given in Equation (2).

One way to define a slightly different concept class is to approximate the non-continuous function, ϕ_{0-1} , with a Lipschitz continuous function, $\phi : \mathbb{R} \rightarrow [0, 1]$, which is often called a transfer function. For example, we can use a sigmoidal transfer function

$$\phi_{\text{sig}}(a) \stackrel{\text{def}}{=} \frac{1}{1 + \exp(-4La)} , \quad (3)$$

which is a L -Lipschitz function. Other L -Lipschitz transfer functions are the erf function and the piece-wise linear function:

$$\phi_{\text{erf}}(a) \stackrel{\text{def}}{=} \frac{1}{2} (1 + \text{erf}(\sqrt{\pi} La)) \quad , \quad \phi_{\text{pw}}(a) \stackrel{\text{def}}{=} \max \{ \min \{ \frac{1}{2} + La, 1 \}, 0 \} \quad (4)$$

An illustration of these transfer functions is given in Figure 1. Analogously to the definition of $H_{\phi_{0-1}}$, for a general transfer function ϕ we define H_{ϕ} to be the set of predictors $\mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x} \rangle)$. Since now the range of ϕ is not $\{0, 1\}$ but rather the entire interval $[0, 1]$, we interpret $\phi(\langle \mathbf{w}, \mathbf{x} \rangle)$ as the probability to output the label 1. The definition of $\text{err}_{\mathcal{D}}(h)$ remains¹ as in Equation (1).

The advantage of using a Lipschitz transfer function can be seen via Rademacher generalization bounds [3]. In fact, a simple corollary of the contraction lemma implies the following:

Theorem 1 *Let $\epsilon, \delta \in (0, 1)$ and let ϕ be an L -Lipschitz transfer function. Let m be an integer satisfying*

$$m \geq \left(\frac{2L + 3\sqrt{2 \ln(8/\delta)}}{\epsilon} \right)^2 .$$

Then, for any distribution \mathcal{D} over $\mathcal{X} \times \{0, 1\}$, the ERM algorithm (ϵ, δ) -learns the concept class H_{ϕ} using m examples.

¹Note that in this case $\text{err}_{\mathcal{D}}(h)$ can be interpreted as $\mathbb{P}_{(\mathbf{x}, y) \sim \mathcal{D}, b \sim \phi(\langle \mathbf{w}, \mathbf{x} \rangle)} [y \neq b]$.

The above theorem tells us that the sample complexity of learning H_ϕ is $\tilde{\Omega}(L^2/\epsilon^2)$. Crucially, the sample complexity does not depend on the dimensionality of \mathcal{X} , but only on the Lipschitz constant of the transfer function. This allows us to learn with kernels, when the dimensionality of \mathcal{X} can even be infinite. A related analysis compares the error rate of a halfspace \mathbf{w} to the number of margin mistakes \mathbf{w} makes on the training set - see Section 4.1 for a comparison.

From the computational complexity point of view, the result given in Theorem 1 is problematic, since the ERM algorithm should solve the non-convex optimization problem

$$\operatorname{argmin}_{\mathbf{w}: \|\mathbf{w}\| \leq 1} \frac{1}{m} \sum_{i=1}^m |\phi(\langle \mathbf{w}, \mathbf{x}_i \rangle) - y_i|. \quad (5)$$

Solving this problem in polynomial time is hard under reasonable assumptions (see Section 3 in which we present a formal hardness result). Adapting a technique due to [6] we show in the full version of this paper [22] that it is possible to find an ϵ -accurate solution to Equation (5) (where the transfer function is ϕ_{pw}) in time $\text{poly}\left(\exp\left(\frac{L^2}{\epsilon^2} \log\left(\frac{L}{\epsilon}\right)\right)\right)$. The main contribution of this paper is the derivation and analysis of a more simple learning algorithm that (ϵ, δ) -learns the class H_{sig} using time and sample complexity of at most $\text{poly}\left(\exp\left(L \log\left(\frac{L}{\epsilon}\right)\right)\right)$. That is, the runtime of our algorithm is exponentially smaller than the runtime required to solve the ERM problem using the technique described in [6]. Moreover, the algorithm of [6] performs an exhaustive search over all $(L/\epsilon)^2$ subsets of the m examples in the training set, and therefore its runtime is always order of m^{L^2/ϵ^2} . In contrast, our algorithm's runtime depends on a parameter B , which is bounded by $\exp(L)$ only under a worst-case assumption. Depending on the underlying distribution, B can be much smaller than the worst-case bound. In practice, we will cross-validate for B , and therefore the worst-case bound will often be pessimistic.

The rest of the paper is organized as follows. In Section 2 we describe our main results. Next, in Section 3 we provide a hardness result, showing that it is not likely that there exists an algorithm that learns H_{sig} or H_{pw} in time polynomial in L . We outline additional related work in Section 4. In particular, the relation between our approach and margin-based analysis is described in Section 4.1, and the relation to approaches utilizing a distributional assumption is discussed in Section 4.2. We wrap up with a discussion in Section 5.

2 Main Results

In this section we present our main result. Recall that we would like to derive an algorithm which learns the class H_{sig} . However, the ERM optimization problem associated with H_{sig} is non-convex. The main idea behind our construction is to learn a larger hypothesis class, denoted H_B , which approximately contains H_{sig} , and for which the ERM optimization problem becomes convex. The price we need to pay is that from the statistical point of view, it is more difficult to learn the class H_B than the class H_{sig} , therefore the sample complexity increases.

The class H_B we use is a class of *linear* predictors in some other RKHS. The kernel function that implements the inner product in the newly constructed RKHS is

$$K(\mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} \frac{1}{1 - \nu \langle \mathbf{x}, \mathbf{x}' \rangle}, \quad (6)$$

where $\nu \in (0, 1)$ is a parameter and $\langle \mathbf{x}, \mathbf{x}' \rangle$ is the inner product in the original RKHS. As mentioned previously, $\langle \mathbf{x}, \mathbf{x}' \rangle$ is usually implemented by some kernel function $K'(\mathbf{z}, \mathbf{z}')$, where \mathbf{z} and \mathbf{z}' are the pre-images of \mathbf{x} and \mathbf{x}' with respect to the feature mapping induced by K' . Therefore, the kernel in Equation (6) is simply a composition with K' , i.e. $K(\mathbf{z}, \mathbf{z}') = 1/(1 - \nu K'(\mathbf{z}, \mathbf{z}'))$.

To simplify the presentation we will set $\nu = 1/2$, although in practice other choices might be more effective. It is easy to verify that K is a valid positive definite kernel function (see for example [19, 10]). Therefore, there exists some mapping $\psi : \mathcal{X} \rightarrow \mathbb{V}$, where \mathbb{V} is an RKHS with $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle = K(\mathbf{x}, \mathbf{x}')$. The class H_B is defined to be:

$$H_B \stackrel{\text{def}}{=} \{\mathbf{x} \mapsto \langle \mathbf{v}, \psi(\mathbf{x}) \rangle : \mathbf{v} \in \mathbb{V}, \|\mathbf{v}\|^2 \leq B\}. \quad (7)$$

The main result we prove in this section is the following:

Theorem 2 *Let $\epsilon, \delta \in (0, 1)$ and let $L \geq 3$. Let $B = 2L^4 + \exp\left(7L \log\left(\frac{2L}{\epsilon}\right) + 3\right)$ and let m be a sample size that satisfies $m \geq \frac{8B}{\epsilon^2} \left(2 + 9\sqrt{\ln(8/\delta)}\right)^2$. Then, for any distribution \mathcal{D} , with probability of at least $1 - \delta$, any ERM predictor $\hat{h} \in H_B$ with respect to H_B satisfies*

$$\text{err}_{\mathcal{D}}(\hat{h}) \leq \min_{h \in H_{\text{sig}}} \text{err}_{\mathcal{D}}(h_{\text{sig}}) + \epsilon.$$

We note that the bound on B is far from being the tightest possible in terms of constants and second-order terms. Also, the assumption of $L \geq 3$ is rather arbitrary, and is meant to simplify the presentation of the bound.

To prove this theorem, we start with analyzing the time and sample complexity of learning H_B . The sample complexity analysis follows directly from a Rademacher generalization bound [3]. In particular, the following theorem tells us that the sample complexity of learning H_B with the ERM rule is order of B/ϵ^2 examples.

Theorem 3 *Let $\epsilon, \delta \in (0, 1)$, let $B \geq 1$, and let m be a sample size that satisfies*

$$m \geq \frac{2B}{\epsilon^2} \left(2 + 9\sqrt{\ln(8/\delta)}\right)^2 .$$

Then, for any distribution \mathcal{D} , the ERM algorithm (ϵ, δ) -learns H_B .

Proof Since $K(\mathbf{x}, \mathbf{x}) \leq 2$, the Rademacher complexity of H_B is bounded by $\sqrt{2B/m}$ (see also [13]). Additionally, using Cauchy-Schwartz inequality we have that the loss is bounded, $|\langle \mathbf{v}, \psi(\mathbf{x}) \rangle - y| \leq \sqrt{2B} + 1$. The result now follows directly from [3, 13]. ■

Next, we show that the ERM problem with respect to H_B can be solved in time $\text{poly}(m)$. The ERM problem associated with H_B is

$$\min_{\mathbf{v}: \|\mathbf{v}\|^2 \leq B} \frac{1}{m} \sum_{i=1}^m |\langle \mathbf{v}, \psi(\mathbf{x}_i) \rangle - y_i| .$$

Since the objective function is defined only via inner products with $\psi(\mathbf{x}_i)$, and the constraint on \mathbf{v} is defined by the ℓ_2 -norm, it follows by the Representer theorem [26] that there is an optimal solution \mathbf{v}^* that can be written as $\mathbf{v}^* = \sum_{i=1}^m \alpha_i \psi(\mathbf{x}_i)$. Therefore, instead of optimizing over \mathbf{v} , we can optimize over the set of weights $\alpha_1, \dots, \alpha_m$ by solving the equivalent optimization problem

$$\min_{\alpha_1, \dots, \alpha_m} \frac{1}{m} \sum_{i=1}^m \left| \sum_{j=1}^m \alpha_j K(\mathbf{x}_j, \mathbf{x}_i) - y_i \right| \quad \text{s.t.} \quad \sum_{i,j=1}^m \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \leq B .$$

This is a convex optimization problem in \mathbb{R}^m and therefore can be solved in time $\text{poly}(m)$ using standard optimization tools.² We therefore obtain:

Corollary 1 *Let $\epsilon, \delta \in (0, 1)$ and let $B \geq 1$. Then, for any distribution \mathcal{D} , it is possible to (ϵ, δ) -learn H_B in sample and time complexity of $\text{poly}\left(\frac{B}{\epsilon} \log(1/\delta)\right)$.*

It is left to understand why the class H_B approximately contains the class H_{sig} . Recall that for any transfer function, ϕ , we define the class H_ϕ to be all the predictors of the form $\mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x} \rangle)$. The first step is to show that H_B contains the union of H_ϕ over all polynomial transfer functions that satisfy a certain boundedness condition on their coefficients.

Lemma 1 *Let P_B be the following set of polynomials (possibly with infinite degree)*

$$P_B \stackrel{\text{def}}{=} \left\{ p(a) = \sum_{j=0}^{\infty} \beta_j a^j : \sum_{j=0}^{\infty} \beta_j^2 2^j \leq B \right\} . \quad (8)$$

Then,

$$\bigcup_{p \in P_B} H_p \subset H_B .$$

Proof To simplify the proof, we first assume that \mathcal{X} is simply the unit ball in \mathbb{R}^n , for an arbitrarily large but finite n . Consider the mapping $\psi : \mathcal{X} \rightarrow \mathbb{R}^{\mathbb{N}}$ defined as follows: for any $\mathbf{x} \in \mathcal{X}$, we let $\psi(\mathbf{x})$ be an infinite vector, indexed by k_1, \dots, k_j for all $(k_1, \dots, k_j) \in \{1, \dots, n\}^j$ and $j = 0 \dots \infty$, where

²In fact, using stochastic gradient descent, we can (ϵ, δ) -learn H_B in time $O(m^2)$, where m is as defined in Theorem 3 —See for example [8, 20].

the entry at index k_1, \dots, k_j equals $2^{-j/2} x_{k_1} \cdot x_{k_2} \cdots x_{k_j}$. The inner-product between $\psi(\mathbf{x})$ and $\psi(\mathbf{x}')$ for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ can be calculated as follows,

$$\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle = \sum_{j=0}^{\infty} \sum_{(k_1, \dots, k_j) \in \{1, \dots, n\}^j} 2^{-j} x_{k_1} x'_{k_1} \cdots x_{k_j} x'_{k_j} = \sum_{j=0}^{\infty} 2^{-j} (\langle \mathbf{x}, \mathbf{x}' \rangle)^j = \frac{1}{1 - \frac{1}{2} \langle \mathbf{x}, \mathbf{x}' \rangle}.$$

This is exactly the kernel function defined in Equation (6) (recall that we set $\nu = 1/2$) and therefore ψ maps to the RKHS defined by K . Consider any polynomial $p(a) = \sum_{j=0}^{\infty} \beta_j a^j$ in P_B , and any $\mathbf{w} \in \mathcal{X}$. Let $\mathbf{v}_{\mathbf{w}}$ be an element in $\mathbb{R}^{\mathbb{N}}$ explicitly defined as being equal to $\beta_j 2^{j/2} w_{k_1} \cdots w_{k_j}$ at index k_1, \dots, k_j (for all $k_1, \dots, k_j \in \{1, \dots, n\}^j, j = 0 \dots \infty$). By definition of ψ and $\mathbf{v}_{\mathbf{w}}$, we have that

$$\langle \mathbf{v}_{\mathbf{w}}, \psi(\mathbf{x}) \rangle = \sum_{j=0}^{\infty} \sum_{k_1, \dots, k_j} 2^{-j/2} \beta_j 2^{j/2} w_{k_1} \cdots w_{k_j} x_{k_1} \cdots x_{k_j} = \sum_{j=0}^{\infty} \beta_j (\langle \mathbf{w}, \mathbf{x} \rangle)^j = p(\langle \mathbf{w}, \mathbf{x} \rangle).$$

In addition,

$$\|\mathbf{v}_{\mathbf{w}}\|^2 = \sum_{j=0}^{\infty} \sum_{k_1, \dots, k_j} \beta_j^2 2^j w_{k_1}^2 \cdots w_{k_j}^2 = \sum_{j=0}^{\infty} \beta_j^2 2^j \sum_{k_1} w_{k_1}^2 \sum_{k_2} w_{k_2}^2 \cdots \sum_{k_j} w_{k_j}^2 = \sum_{j=0}^{\infty} \beta_j^2 2^j (\|\mathbf{w}\|^2)^j \leq B.$$

Thus, the predictor $\mathbf{x} \mapsto \langle \mathbf{v}_{\mathbf{w}}, \psi(\mathbf{x}) \rangle$ belongs to H_B and is the same as the predictor $\mathbf{x} \mapsto p(\langle \mathbf{w}, \mathbf{x} \rangle)$. This proves that $H_p \subset H_B$ for all $p \in P_B$ as required. Finally, if \mathcal{X} is an infinite dimensional RKHS, the only technicality is that in order to represent \mathbf{x} as a (possibly infinite) vector, we need to show that our RKHS has a countable basis. This holds since the inner product $\langle \mathbf{x}, \mathbf{x}' \rangle$ over \mathcal{X} is continuous and bounded (see [1]). \blacksquare

Finally, the following lemma states that with a sufficiently large B , there exists a polynomial in P_B which approximately equals to ϕ_{sig} . This implies that H_B approximately contains H_{sig} .

Lemma 2 *Let ϕ_{sig} be as defined in Equation (3), where for simplicity we assume $L \geq 3$. For any $\epsilon > 0$, let*

$$B = 2L^4 + \exp(7L \log(\frac{2L}{\epsilon}) + 3).$$

Then there exists $p \in P_B$ such that

$$\forall \mathbf{x}, \mathbf{w} \in \mathcal{X}, \quad |p(\langle \mathbf{w}, \mathbf{x} \rangle) - \phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle)| \leq \epsilon.$$

The proof of the lemma is based on a Chebyshev approximation technique and is given in the full version of our paper [22]. Since the proof is rather involved, we also present a similar lemma, whose proof is simpler, for the ϕ_{erf} transfer function (see [22]). It is interesting to note that ϕ_{erf} actually *belongs* to P_B for a sufficiently large B , since it can be defined via its infinite-degree Taylor expansion. However, the bound for ϕ_{erf} depends on $\exp(L^2)$, rather than $\exp(L)$ for the sigmoid transfer function ϕ_{sig} .

Finally, Theorem 2 is obtained as follows: Combining Theorem 3 and Lemma 1 we get that with probability of at least $1 - \delta$,

$$\text{err}_{\mathcal{D}}(\hat{h}) \leq \min_{h \in H_B} \text{err}_{\mathcal{D}}(h) + \epsilon/2 \leq \min_{p \in P_B} \min_{h \in H_p} \text{err}_{\mathcal{D}}(h) + \epsilon/2. \quad (9)$$

From Lemma 2 we obtain that for any $\mathbf{w} \in \mathcal{X}$, if $h(\mathbf{x}) = \phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle)$ then there exists a polynomial $p_0 \in P_B$ such that if $h'(\mathbf{x}) = p_0(\langle \mathbf{w}, \mathbf{x} \rangle)$ then $\text{err}_{\mathcal{D}}(h') \leq \text{err}_{\mathcal{D}}(h) + \epsilon/2$. Since it holds for all \mathbf{w} , we get that

$$\min_{p \in P_B} \min_{h \in H_p} \text{err}_{\mathcal{D}}(h) \leq \min_{h \in H_{\text{sig}}} \text{err}_{\mathcal{D}}(h) + \epsilon/2.$$

Combining this with Equation (9), Theorem 2 follows.

3 Hardness

In this section we derive a hardness result for agnostic learning of H_{sig} or H_{pw} with respect to the zero-one loss. The hardness result relies on the hardness of standard (non-agnostic)³ PAC learning of intersection of halfspaces given in Klivans and Sherstov [16] (see also similar arguments in [11]). The hardness result is representation-independent — it makes no restrictions on the learning algorithm and in particular also holds for improper learning algorithms. The hardness result is based on the following cryptographic assumption:

³In the *standard* PAC model, we assume that some hypothesis in the class has $\text{err}_{\mathcal{D}}(h) = 0$, while in the *agnostic* PAC model, which we study in this paper, $\text{err}_{\mathcal{D}}(h)$ might be strictly greater than zero for all $h \in H$. Note that our definition of (ϵ, δ) -learning in this paper is in the agnostic model.

Assumption 1 *There is no polynomial time solution to the $\tilde{O}(n^{1.5})$ -unique-Shortest-Vector-Problem.*

In a nutshell, given a basis $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{R}^n$, the $\tilde{O}(n^{1.5})$ -unique-Shortest-Vector-Problem consists of finding the shortest nonzero vector in $\{a_1\mathbf{v}_1 + \dots + a_n\mathbf{v}_n : a_1, \dots, a_n \in \mathcal{Z}\}$, even given the information that it is shorter by a factor of at least $\tilde{O}(n^{1.5})$ than any other non-parallel vector. This problem is believed to be hard - there are no known sub-exponential algorithms, and it is known to be NP-hard if $\tilde{O}(n^{1.5})$ is replaced by a small constant (see [16] for more details).

With this assumption, Klivans and Sherstov proved the following:

Theorem 4 (Theorem 1.2 in Klivans and Sherstov [16]) *Let $\mathcal{X} = \{\pm 1\}^n$, let*

$$H = \{\mathbf{x} \mapsto \phi_{0,1}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2) : \theta \in \mathbb{N}, \mathbf{w} \in \mathbb{N}^n, |\theta| + \|\mathbf{w}\|_1 \leq \text{poly}(n)\},$$

and let $H_k = \{\mathbf{x} \mapsto (h_1(\mathbf{x}) \wedge \dots \wedge h_k(\mathbf{x})) : \forall i, h_i \in H\}$. Then, based on Assumption 1, H_k is not efficiently learnable in the standard PAC model for any $k = n^\rho$ where $\rho > 0$ is a constant.

The above theorem implies the following.

Lemma 3 *Based on Assumption 1, there is no algorithm that runs in time $\text{poly}(n, 1/\epsilon, 1/\delta)$ and (ϵ, δ) -learns the class H defined in Theorem 4.*

Proof To prove the lemma we show that if there is a polynomial time algorithm that learns H in the *agnostic* model, then there exists a weak learning algorithm (with a polynomial edge) that learns H_k in the standard (non-agnostic) PAC model. In the standard PAC model, weak learning implies strong learning [18], hence the existence of a weak learning algorithm that learns H_k will contradict Theorem 4.

Indeed, let \mathcal{D} be any distribution such that there exists $h^* \in H_k$ with $\text{err}_{\mathcal{D}}(h^*) = 0$. Let us rewrite $h^* = h_1^* \wedge \dots \wedge h_k^*$ where for all i , $h_i^* \in H$. To show that there exists a weak learner, we first show that there exists some $h \in H$ with $\text{err}_{\mathcal{D}}(h) \leq 1/2 - 1/n$.

Since for each \mathbf{x} if $h^*(\mathbf{x}) = 0$ then there exists j s.t. $h_j^*(\mathbf{x}) = 0$, we can use the union bound to get that

$$1 = \mathbb{P}[\exists j : h_j^*(\mathbf{x}) = 0 | h^*(\mathbf{x}) = 0] \leq \sum_j \mathbb{P}[h_j^*(\mathbf{x}) = 0 | h^*(\mathbf{x}) = 0] \leq k \max_j \mathbb{P}[h_j^*(\mathbf{x}) = 0 | h^*(\mathbf{x}) = 0].$$

So, for j that maximizes $\mathbb{P}[h_j^*(\mathbf{x}) = 0 | h^*(\mathbf{x}) = 0]$ we get that $\mathbb{P}[h_j^*(\mathbf{x}) = 0 | h^*(\mathbf{x}) = 0] \geq 1/k$. Therefore,

$$\begin{aligned} \text{err}_{\mathcal{D}}(h_j^*) &= \mathbb{P}[h_j^*(\mathbf{x}) = 1 \wedge h^*(\mathbf{x}) = 0] = \mathbb{P}[h^*(\mathbf{x}) = 0] \mathbb{P}[h_j^*(\mathbf{x}) = 1 | h^*(\mathbf{x}) = 0] \\ &= \mathbb{P}[h^*(\mathbf{x}) = 0] (1 - \mathbb{P}[h_j^*(\mathbf{x}) = 0 | h^*(\mathbf{x}) = 0]) \leq \mathbb{P}[h^*(\mathbf{x}) = 0] (1 - 1/k). \end{aligned}$$

Now, if $\mathbb{P}[h^*(\mathbf{x}) = 0] \leq (1/2 + 1/k)$ then the above gives

$$\text{err}_{\mathcal{D}}(h_j^*) \leq (1/2 + 1/k)(1 - 1/k) = 1/2 + 1/k - 1/(2k) - 1/k^2 \leq 1/2 - 1/k.$$

Otherwise, if $\mathbb{P}[h^*(\mathbf{x}) = 0] > (1/2 + 1/k)$, then the constant predictor $h(\mathbf{x}) = 0$ have $\text{err}_{\mathcal{D}}(h) < 1/2 - 1/k$. In both cases we have shown that there exists a predictor in H with error of at most $1/2 - 1/k$.

Finally, if we can agnostically learn H in time $\text{poly}(n, 1/\epsilon, 1/\delta)$, then we can find h' with $\text{err}_{\mathcal{D}}(h') \leq \min_{h \in H} \text{err}_{\mathcal{D}}(h) + \epsilon \leq 1/2 - 1/k + \epsilon$ in time $\text{poly}(n, 1/\epsilon, 1/\delta)$. This means that we can have a weak learner that runs in polynomial time, and this concludes our proof. \blacksquare

Let h be a hypothesis in the class H defined in Theorem 4 and take any $\mathbf{x} \in \{\pm 1\}^n$. Then, there exist an integer θ and a vector of integers \mathbf{w} such that $h(\mathbf{x}) = \phi_{0,1}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2)$. But since $\langle \mathbf{w}, \mathbf{x} \rangle - \theta$ is also an integer, if we let $L = 1$ this means that $h(\mathbf{x}) = \phi_{\text{pw}}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2)$ as well. Furthermore, letting $\mathbf{x}' \in \mathbb{R}^{n+1}$ denote the concatenation of \mathbf{x} with the constant 1 and letting $\mathbf{w}' \in \mathbb{R}^{n+1}$ denote the concatenation of \mathbf{w} with the scalar $(-\theta - 1/2)$ we obtain that $h(\mathbf{x}) = \phi_{\text{pw}}(\langle \mathbf{w}', \mathbf{x}' \rangle)$. Last, let us normalize $\tilde{\mathbf{w}} = \mathbf{w}' / \|\mathbf{w}'\|$, $\tilde{\mathbf{x}} = \mathbf{x}' / \|\mathbf{x}'\|$, and redefine L to be $\|\mathbf{w}'\| \|\mathbf{x}'\|$, we get that $h(\mathbf{x}) = \phi_{\text{pw}}(\langle \tilde{\mathbf{w}}, \tilde{\mathbf{x}} \rangle)$. That is, we have shown that H is contained in a class of the form H_{pw} with a Lipschitz constant bounded by $\text{poly}(n)$. Combining the above with Lemma 3 we obtain the following:

Corollary 2 *Let L be a Lipschitz constant and let H_{pw} be the class defined by the L -Lipschitz transfer function ϕ_{pw} . Then, based on Assumption 1, there is no algorithm that runs in time $\text{poly}(L, 1/\epsilon, 1/\delta)$ and (ϵ, δ) -learns the class H_{pw} .*

A similar argument leads to the hardness of learning H_{sig} .

Theorem 5 *Let L be a Lipschitz constant and let H_{sig} be the class defined by the L -Lipschitz transfer function ϕ_{sig} . Then, based on Assumption 1, there is no algorithm that runs in time $\text{poly}(L, 1/\epsilon, 1/\delta)$ and (ϵ, δ) -learns the class H_{sig} .*

Proof Let h be a hypothesis in the class H defined in Theorem 4 and take any $\mathbf{x} \in \{\pm 1\}^n$. Then, there exist an integer θ and a vector of integers \mathbf{w} such that $h(\mathbf{x}) = \phi_{0,1}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2)$. However, since $\langle \mathbf{w}, \mathbf{x} \rangle - \theta$ is also an integer, we see that

$$|\phi_{0,1}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2) - \phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2)| \leq \frac{1}{1 + \exp(2L)} .$$

This means that for any $\epsilon > 0$, if we pick $L = \frac{\log(2/\epsilon - 1)}{2}$ and define $h_{\text{sig}}(\mathbf{x}) = \phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2)$, then $|h(\mathbf{x}) - h_{\text{sig}}(\mathbf{x})| \leq \epsilon/2$. Furthermore, letting $\mathbf{x}' \in \mathbb{R}^{n+1}$ denote the concatenation of \mathbf{x} with the constant 1 and letting $\mathbf{w}' \in \mathbb{R}^{n+1}$ denote the concatenation of \mathbf{w} with the scalar $(-\theta - 1/2)$ we obtain that $h_{\text{sig}}(\mathbf{x}) = \phi_{\text{sig}}(\langle \mathbf{w}', \mathbf{x}' \rangle)$. Last, let us normalize $\tilde{\mathbf{w}} = \mathbf{w}'/\|\mathbf{w}'\|$, $\tilde{\mathbf{x}} = \mathbf{x}'/\|\mathbf{x}'\|$, and redefine L to be

$$L = \frac{\|\mathbf{w}'\| \|\mathbf{x}'\| \log(2/\epsilon - 1)}{2} \tag{10}$$

so that $h_{\text{sig}}(\mathbf{x}) = \phi_{\text{sig}}(\langle \tilde{\mathbf{w}}, \tilde{\mathbf{x}} \rangle)$. Thus we see that if there exists an algorithm that runs in time $\text{poly}(L, 1/\epsilon, 1/\delta)$ and $(\epsilon/2, \delta)$ -learns the class H_{sig} , then since for all $h \in H$ exists $h_{\text{sig}} \in H_{\text{sig}}$ such that $|h_{\text{sig}}(\mathbf{x}) - h(\mathbf{x})| \leq \epsilon/2$, there also exists an algorithm that (ϵ, δ) -learns the concept class H defined in Theorem 4 in time polynomial in $(L, 1/\epsilon, 1/\delta)$ (for L defined in Equation 10). But by definition of L in Equation 10 and the fact that $\|\mathbf{w}'\|$ and $\|\mathbf{x}'\|$ are of size $\text{poly}(n)$, this means that there is an algorithm that runs in time polynomial in $(n, 1/\epsilon, 1/\delta)$ and (ϵ, δ) -learns the class H , which contradicts Lemma 3. \blacksquare

4 Related work

The problem of learning kernel-based halfspaces has been extensively studied before, mainly in the framework of SVM [25, 10, 19]. When the data is separable with a margin μ , it is possible to learn a halfspaces in polynomial time. The learning problem becomes much more difficult when the data is not separable with margin.

In terms of hardness results, [6] derive hardness results for proper learning with sufficiently small margins. There are also strong hardness of approximation results for *proper* learning *without* margin (see for example [12] and the references therein). We emphasize that we allow improper learning, which is just as useful for the purpose of learning good classifiers, and thus these hardness results do not apply. Instead, the hardness result we derived in Section 3 hold for improper learning as well. As mentioned before, the main tool we rely on for deriving the hardness result is the representation independent hardness result for learning intersections of halfspaces given in [16].

Practical algorithms such as SVM often replace the 0-1 error function with a convex surrogate, and then apply convex optimization tools. However, there are no guarantees on how well the surrogate function approximates the 0-1 error function. Recently, [27, 4] studied the *asymptotic* relationship between surrogate convex loss functions and the 0-1 error function. In contrast, in this paper we show that even with a finite sample, surrogate convex loss functions can be competitive with the 0-1 error function as long as we replace inner-products with the kernel $K(\mathbf{x}, \mathbf{x}') = 1/(1 - 0.5\langle \mathbf{x}, \mathbf{x}' \rangle)$.

4.1 Margin analysis

Recall that we circumvented the dependence of the VC dimension of $H_{\phi_{0-1}}$ on the dimensionality of \mathcal{X} by replacing ϕ_{0-1} with a Lipschitz transfer function. Another common approach is to require that the learned classifier will be competitive with the *margin* error rate of the optimal halfspace. Formally, the μ -margin error rate of a halfspace of the form $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{1}(\langle \mathbf{w}, \mathbf{x} \rangle > 0)$ is defined as:

$$\text{err}_{\mathcal{D}, \mu}(\mathbf{w}) = \Pr[h_{\mathbf{w}}(\mathbf{x}) \neq y \vee |\langle \mathbf{w}, \mathbf{x} \rangle| \leq \mu] . \tag{11}$$

Intuitively, $\text{err}_{\mathcal{D}, \mu}(\mathbf{w})$ is the error rate of $h_{\mathbf{w}}$ had we μ -shifted each point in the worst possible way. Margin based analysis restates the goal of the learner (as given in Equation (2)) and requires that the learner will find a classifier h that satisfies:

$$\text{err}_{\mathcal{D}}(h) \leq \min_{\mathbf{w}: \|\mathbf{w}\|=1} \text{err}_{\mathcal{D}, \mu}(\mathbf{w}) + \epsilon . \tag{12}$$

Bounds of the above form are called margin-based bounds and are widely used in the statistical analysis of Support Vector Machines and AdaBoost. It was shown [3, 17] that $m = \Theta(\log(1/\delta)/(\mu\epsilon)^2)$ examples are sufficient (and necessary) to learn a classifier for which Equation (12) holds with probability of at least $1 - \delta$. Note that as in the sample complexity bound we gave in Theorem 1, the margin based sample complexity bound also does not depend on the dimension.

In fact, the Lipschitz approach used in this paper and the margin-based approach are closely related. First, it is easy to verify that if we set $L = 1/(2\mu)$, then for any \mathbf{w} the hypothesis $h(\mathbf{x}) = \phi_{\text{pw}}(\langle \mathbf{w}, \mathbf{x} \rangle)$ satisfies $\text{err}_{\mathcal{D}}(h) \leq \text{err}_{\mathcal{D},\mu}(\mathbf{w})$. Therefore, an algorithm that (ϵ, δ) -learns H_{pw} also guarantees that Equation (12) holds. Second, it is also easy to verify that if we set $L = \frac{1}{4\mu} \log\left(\frac{2-\epsilon}{\epsilon}\right)$ then for any \mathbf{w} the hypothesis $h(\mathbf{x}) = \phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle)$ satisfies $\text{err}_{\mathcal{D}}(h) \leq \text{err}_{\mathcal{D},\mu}(\mathbf{w}) + \epsilon/2$. Therefore, an algorithm that $(\epsilon/2, \delta)$ -learns H_{sig} also guarantees that Equation (12) holds.

As a direct corollary of the above discussion we obtain that it is possible to learn a vector \mathbf{w} that guarantees Equation (12) in time $\text{poly}(\exp(\tilde{O}(1/\mu)))$.

A computational complexity analysis under margin assumptions was first carried out in [6] (see also the hierarchical worst-case analysis recently proposed in [5]). The technique used in [6] is based on the observation that in the noise-free case, an optimal halfspace can be expressed as a linear sum of at most $1/\mu^2$ examples. Therefore, one can perform an exhaustive search over all sub-sequences of $1/\mu^2$ examples, and choose the optimal halfspace. Note that this algorithm will always run in time m^{1/μ^2} . Since the sample complexity bound requires that m will be order of $1/(\mu\epsilon)^2$, the runtime of the method described by [6] becomes $\text{poly}(\exp(\tilde{O}(1/\mu^2)))$. In comparison, our algorithm achieves a better runtime of $\text{poly}(\exp(\tilde{O}(1/\mu)))$. Moreover, while the algorithm of [6] performs an exhaustive search, our algorithm’s runtime depends on the parameter B , which is $\text{poly}(\exp(\tilde{O}(1/\mu)))$ only under a worst-case assumption. Since in practice we will cross-validate for B , it is plausible that in many real-world scenarios the runtime of our algorithm will be much smaller.

4.2 Distributional Assumptions

The idea of approximating the zero-one transfer function with a polynomial was first proposed by [14] who studied the problem of agnostically learning halfspaces without kernels in \mathbb{R}^n under distributional assumption. In particular, they showed that if the distribution over \mathcal{X} is uniform over the unit ball, then it is possible to agnostically learn $H_{\phi_{0-1}}$ in time $\text{poly}(n^{1/\epsilon^4})$. This was further generalized by [7], who showed that similar bounds hold for product distributions.

Beside distributional assumptions, these works are characterized by explicit dependence on the dimension of \mathcal{X} , and therefore are not adequate for the kernel-based setting we consider in this paper, in which the dimensionality of \mathcal{X} can even be infinite. More precisely, while [14] try to approximate the zero-one transfer function with a low-degree polynomial, we require instead that the coefficients of the polynomials are bounded. The principle that when learning in high dimensions “the size of the parameters is more important than their number” was one of the main advantages in the analysis of the statistical properties of several learning algorithms (e.g. [2]).

Interestingly, in [21] we show that the very same algorithm we use in this paper recover the same complexity bound of [14].

5 Discussion

In this paper we described and analyzed a new technique for agnostically learning kernel-based halfspaces with the zero-one loss function. The bound we derive has an exponential dependence on L , the Lipschitz coefficient of the transfer function. While we prove that (under a certain cryptographic assumption) no algorithm can have a polynomial dependence on L , the immediate open question is whether the dependence on L can be further improved.

A perhaps surprising property of our analysis is that we propose a single algorithm, returning a single classifier, which is simultaneously competitive against *all* transfer functions $p \in P_B$. In particular, it learns with respect to the “optimal” transfer function, where by optimal we mean the one which attains the smallest error rate, $\mathbb{E}[|p(\langle \mathbf{w}, \mathbf{x} \rangle) - y|]$, over the distribution \mathcal{D} .

Our algorithm boils down to linear regression with the absolute loss function and while composing a particular kernel function over our original RKHS. It is possible to show that solving the vanilla SVM, with the hinge-loss, and composing again our particular kernel over the desired kernel, can also give similar guarantees. It is therefore interesting to study if there is something special about the kernel we propose or maybe other kernel functions (e.g. the Gaussian kernel) can give similar guarantees.

Another possible direction is to consider other types of margin-based analysis or transfer functions. For example, in the statistical learning literature, there are several definitions of “noise”

conditions, some of them are related to margin, which lead to faster decrease of the error rate as a function of the number of examples (see for example [9, 24, 23]). Studying the computational complexity of learning under these conditions is left to future work.

Acknowledgments

We would like to thank Adam Klivans for helping with the Hardness results. This work was partially supported by a Google Faculty Research Grant.

References

- [1] C. Thomas-Agnan A. Berlinet. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer, 2003.
- [2] P. L. Bartlett. For valid generalization, the size of the weights is more important than the size of the network. In *Advances in Neural Information Processing Systems 9*, 1997.
- [3] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.
- [4] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101:138–156, 2006.
- [5] S. Ben-David. Alternative measures of computational complexity. In *TAMC*, 2006.
- [6] S. Ben-David and H. Simon. Efficient learning of linear perceptrons. In *NIPS*, 2000.
- [7] E. Blais, R. O’Donnell, and K Wimmer. Polynomial regression under arbitrary product distributions. In *COLT*, 2008.
- [8] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, pages 161–168, 2008.
- [9] O. Bousquet. *Concentration Inequalities and Empirical Processes Theory Applied to the Analysis of Learning Algorithms*. PhD thesis, Ecole Polytechnique, 2002.
- [10] N. Cristianini and J. Shawe-Taylor. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
- [11] V. Feldman, P. Gopalan, S. Khot, and A.K. Ponnuswami. New results for learning noisy parities and halfspaces. In *In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.
- [12] V. Guruswami and P. Raghavendra. Hardness of learning halfspaces with noise. In *Proceedings of the 47th Foundations of Computer Science (FOCS)*, 2006.
- [13] S.M. Kakade, K. Sridharan, and A. Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *NIPS*, 2008.
- [14] A. Kalai, A.R. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. In *Proceedings of the 46th Foundations of Computer Science (FOCS)*, 2005.
- [15] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning. In *COLT*, pages 341–352, July 1992. To appear, *Machine Learning*.
- [16] Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. In *FOCS*, 2006.
- [17] D. A. McAllester. Simplified PAC-Bayesian margin bounds. In *COLT*, pages 203–215, 2003.
- [18] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
- [19] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.
- [20] S. Shalev-Shwartz and N. Srebro. SVM optimization: Inverse dependence on training set size. In *International Conference on Machine Learning*, pages 928–935, 2008.

- [21] S. Shalev-Shwartz, O. Shamir, and K. Sridharan. Agnostically learning halfspaces with margin errors. Technical report, Toyota Technological Institute, 2009.
- [22] S. Shalev-Shwartz, O. Shamir, and K. Sridharan. Learning kernel-based halfspaces with the zero-one loss, 2010. Technical Report, available at arXiv:1005.3681.
- [23] I. Steinwart and C. Scovel. Fast rates for support vector machines using gaussian kernels. *Annals of Statistics*, 35:575, 2007.
- [24] A. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32:135–166, 2004.
- [25] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
- [26] G. Wahba. *Spline Models for Observational Data*. SIAM, 1990.
- [27] T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85, 2004.

Ranking with kernels in Fourier space

Risi Kondor

Center for the Mathematics of Information
California Institute of Technology
risi@caltech.edu

Marconi Barbosa

NICTA, Australian National University
marconi.barbosa@nicta.com.au

Abstract

In typical ranking problems the total number n of items to be ranked is relatively large, but each data instance involves only $k \ll n$ items. This paper examines the structure of such partial rankings in Fourier space. Specifically, we develop a kernel-based framework for solving ranking problems, define some canonical kernels on permutations, and show that by transforming to Fourier space, the complexity of computing the kernel between two partial rankings can be reduced from $O((n-k)!^2)$ to $O((2k)^{2k+3})$.

1 Introduction

The word “ranking” covers a wide array of problems from learning a preference function from lists, through fusing the results of multiple search engines to methods for evaluating the results of elections. A closely related problem is “permutation learning” which attempts to learn an assignment between two sets of objects, rather than an ordering of a single set (Helmbold & Warmuth, 2009). On the theory side, “rank aggregation”, i.e., finding a single ranking that best represents a collection of individual rankings according to some metric, is known to be hard, but recently some tractable approximations to this problem have emerged (Ailon et al., 2005).

The common feature of all these problems is that they involve aggregating information from a large set of diverse ranking instances. Typically, the individual instances, rather than being complete rankings of all n items under consideration, are partial rankings that only involve $k \ll n$ items. For example, in the search engine case, we might only take the top ten results from each search engine, giving us a sequence of partially overlapping lists. Similarly, in a movie recommendation system no user will have seen all the movies in our database, and even if they have, they would find it very difficult to give a single linear ordering of a large number of them. Instead, in many situations, including social surveys and election schemes, people are asked to rank order just a small subset of the n items under consideration. An extreme case of this are sports tournaments, such as in chess, where a single global ranking must be established from a large collection of game outcomes, each of which involves only two players.

Ultimately, ranking is a problem of inference on the group of permutations of n objects, called the symmetric group. However, for various mathematical and computational reasons, formulating it as such is difficult. A number of approaches are popular that circumvent this combinatorial problem by reducing ranking to a sequence of binary decisions (Ailon & Mohri, 2008; Balcan et al., 2008), estimating a parametric distribution (Lebanon & Lafferty, 2002), or estimating a scoring function. Depending on the problem at hand, these heuristics might be more or less appropriate. For example, in the search engine case it is reasonable to assume that the relevance of each web page can be described by a real valued score, and, in fact, formulating the ranking problem as learning the score might be a good way to reduce the complexity of the hypothesis space. On the other hand, in the collaborative filtering example the scoring heuristic is not so helpful, since there is no single “best movie”. As a more extreme case, in social surveys there is a lot of interest not just in finding a single consensus ranking, but in finding clusters, trends, and principal axes of variation in the data, which really do force us to treat individual instances as permutations or sets of permutations and take into account the combinatorial structure of the symmetric group.

The algorithmic framework that we use in this paper is that of kernel methods. There are a number of reasons why this is attractive for ranking: (1) kernel methods are flexible in that once we define an appropriate kernel between any two (partial) rankings they allow us to handle diverse data

involving a mixture of ranking instances of different types (partial rankings of different structure and order); (2) by the representer theorem they allow us to circumvent the problem that the symmetric group is of size $n!$; (3) by the input/output kernels formalism they allow us to predict rankings as well as learn from rankings, in fact, there might not even be a distinction between input and output spaces; (4) by the same token, they make it easy to introduce additional features (correlates) that are not rankings; (5) finally, there is a wealth of algorithms to choose from to suit the learning problem at hand.

Despite these potential advantages of kernel methods, they have not been used much in the ranking domain, and to the best of our knowledge the issue of defining generic kernels on the symmetric group has only been addressed in (Kondor, 2008). We believe that one of the main reasons for this is that it is difficult to marginalize kernels to the types of partial rankings that actually occur in data. The main practical conclusion of the present paper is that by a careful analysis in Fourier space, heavily exploiting the underlying algebra of the symmetric group, it is possible to compute such marginalized kernels quite fast.

Our focus in this paper is on structure and computational complexity rather than algorithms, so we will mostly remain agnostic about which kernel algorithm to employ. As a short menu of options we suggest the following.

In the unsupervised setting one can use kernel density estimation or some other similar method to estimate a distribution over rankings. By the representer theorem the resulting distribution takes the form of a linear combination $p(\sigma) = \sum_i \alpha_i K(R_i, \sigma)$, and we can find the “best consensus ranking” by finding $\arg \max_{\sigma \in \mathbb{S}_n} \langle \mathbf{p}, \sigma \rangle$. In political data, to find factors underlying people’s votes one can use kernel PCA. Similarly, to find representative groups of individuals, we would use kernel clustering.

In a more predictive setting, a typical type of question is “Given that user j ranked the movies x_{i_1}, \dots, x_{i_k} in the order $x_{i_1} \succ \dots \succ x_{i_k}$, how would she rank $x_{i'_1}, \dots, x_{i'_\ell}$?” A natural algorithm for this sort of problem is a multi-class SVM that minimizes a regularized risk of the form $\sum_j L(f, R_j) + \langle \mathbf{f}, \mathbf{f} \rangle_K$, where the loss is the multi-class hinge loss comparing the correct ranking of $x_{i'_1}, \dots, x_{i'_\ell}$ conditional on $x_{i_1} \succ \dots \succ x_{i_k}$ to the highest scoring incorrect ranking,

$$L(f, R_j) = \max[0, 1 - (\langle \mathbf{f}, \mathbf{R}_j^e \rangle_K - \max_{\tau \in \mathbb{S}_{k'} \setminus \{e\}} \langle \mathbf{f}, \mathbf{R}_j^\tau \rangle_K)], \quad (1)$$

$\langle \cdot, \cdot \rangle_K$ is the RKHS inner product, \mathbf{R}_j^τ stands for $(x_{i_1} \succ \dots \succ x_{i_k}, x_{i'_{\tau^{-1}(e)}} \succ \dots \succ x_{i'_{\tau^{-1}(1)}})$, which is the j ’th training example, the “outputs” of which have been permuted by τ , and e is the identity permutation. The exact meaning of some of these notations will become clear later in the paper.

This paper has several goals: to define a canonical class of kernels on permutations (Section 3), show how these kernels can be efficiently evaluated in Fourier space (Section 4), and give a detailed analysis of the complexity of the computations involved (Section 5). Our main result is that the inner product between any two k ’th order partial rankings can be computed in at most $(2k)^{2k+3}$ operations, irrespective of the number of items to be ranked (Theorem 13).

To get to this result we need a range of tools from algebra, which we try to present in a way that is as concise yet intuitive as possible. Unfortunately, page limitations prevented us from making the paper entirely self-contained. Background information from representation theory and most proofs had to be relegated to a supplement, which will be published under separate cover.

2 Partial rankings, multirankings, and the group algebra

Let $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ be a set of items to be ranked. We will use $x \succ x'$ to denote that x is ranked higher than x' according to some ranking R . A **total ranking** of \mathcal{X} is then a statement of the form

$$x_{i_1} \succ x_{i_2} \succ \dots \succ x_{i_n}, \quad (2)$$

where i_1, \dots, i_n are distinct indices in $[1, n]$ (in this paper $[i, j]$ denotes $\{i, i+1, \dots, j\}$).

In contrast, a **partial ranking** is a statement of the form $X_1 \succ X_2 \succ \dots \succ X_k$, where X_1, X_2, \dots, X_k are disjoint subsets of \mathcal{X} , and the semantics is that for $i < j$ anything in X_i is ranked above anything in X_j . If a pair of items x and y are both in the same X_i , or one or both of them are not in any of the X_i ’s at all, then their relative order is indeterminate. In other words, neither $x \succ y$, nor $y \succ x$.

Partial rankings can be of different types. For example, in an election example, the vote of each person corresponds to a partial ranking of the form

$$x_{i_1} \succ x_{i_2} \succ \dots \succ x_{i_k}, \quad k \leq n, \quad (3)$$

where, to simplify notation, we abbreviated $\{x_{i_j}\}$ to just x_{i_j} . We call this an **interleaving partial ranking** of order k , because in any total ranking satisfying (3), $x_{i_1}, x_{i_2}, \dots, x_{i_k}$ are interleaved with the other x_i 's. The outcomes of individual games in a chess tournament are an example of this type of partial ranking with $k=2$. In contrast, the top k hits returned by a search engine induce what we call **top- k partial rankings**

$$x_{i_1} \succ x_{i_2} \succ \dots \succ x_{i_k} \succ X_{\text{rest}}, \quad (4)$$

where $X_{\text{rest}} = \mathcal{X} \setminus \{x_{i_1}, \dots, x_{i_k}\}$. Many other types of partial rankings are conceivable. For example, a survey mentioned in (Diaconis, 1988) asked respondents which of 13 characteristics that a child might have is the most desirable, next two most desirable, least desirable, and next two least desirable. This corresponds to a partial ranking of the form $X_1 \succ X_2 \succ X_3 \succ X_4 \succ X_5$, where $|X_1| = |X_5| = 1$, $|X_2| = |X_4| = 2$ and $|X_3| = 7$. While the general framework described in this paper does apply to such more complicated partial rankings, for brevity we will limit our discussion to interleaving and top- k partial rankings.

We will also encounter cases where we have to deal with the conjunction of several partial rankings. We call this a **multiranking**. For example, in a movie recommendation system we might know that a certain user ranked movies x_1, x_2 and x_3 in the order $x_1 \succ x_2 \succ x_3$. Now based on a large number of rankings of similar movies submitted by other users, we want to decide whether given her choice $x_1 \succ x_2 \succ x_3$ she is more likely to enjoy movie x_4 or x_5 . This translates to predicting whether overall the multi-ranking $(x_1 \succ x_2 \succ x_3, x_4 \succ x_5)$ or the multi-ranking $(x_1 \succ x_2 \succ x_3, x_5 \succ x_4)$ is more probable. The most general form of multi-ranking is

$$(X_{1,1} \succ \dots \succ X_{1,k_1}, \dots, X_{\ell,1} \succ \dots \succ X_{\ell,k_\ell}),$$

but in this paper we will only discuss multi-rankings of the form

$$(x_{i_{1,1}} \succ \dots \succ x_{i_{1,k_1}}, \dots, x_{i_{\ell,1}} \succ \dots \succ x_{i_{\ell,k_\ell}}). \quad (5)$$

For simplicity, we will loosely use the term ‘‘partial ranking’’ for both strict partial rankings and multirankings.

2.1 Rankings as sets of permutations

We identify a total ranking, such as (2), with the unique permutation $\sigma: [1, n] \rightarrow [1, n]$ that moves the index of the item ranked first into position n , the index of the item ranked second into position $n-1$, etc.. In other words,

$$x_{\sigma^{-1}(n)} \succ x_{\sigma^{-1}(n-1)} \succ \dots \succ x_{\sigma^{-1}(1)}. \quad (6)$$

While at this point the reversal between ranks and mapping positions might seem like an unnecessary complication, it will simplify the algebra later in the paper.

Partial rankings and multirankings are identified with the set of all permutations that satisfy them. Thus, the interleaving partial ranking (3) corresponds to

$$A_{i_1, \dots, i_k} = \{ \sigma \mid \sigma(i_a) > \sigma(i_b) \text{ if } a < b \},$$

while the top- k partial ranking (4) corresponds to

$$B_{i_1, \dots, i_k} = \{ \sigma \mid \sigma(i_j) = n - j + 1, \forall j \}.$$

Multi-rankings, in general, correspond to the intersection of the sets of permutations associated with their constituents. In particular, the set corresponding to (5) is $C_{i_{1,1} \dots i_{\ell, k_\ell}} = \bigcap_{j=1}^{\ell} A_{i_{j,1}, \dots, i_{j, k_j}}$.

2.2 Rankings as vectors in $\mathbb{C}[\mathbb{S}_n]$

One of the key ideas of the present paper is to identify each (total, partial, or multi-) ranking with a vector in a space called the group algebra of the symmetric group. Let us now clarify this statement.

If we define the product of one permutation σ_1 with another permutation σ_2 as their composition $\sigma_2 \circ \sigma_1$, then with respect to this operation the $n!$ possible permutations of $[1, n]$ form a group. This group is called the **symmetric group** of degree n , and is denoted \mathbb{S}_n . Since \mathbb{S}_n is a **non-commutative** group, in general, $\sigma_1 \sigma_2 \neq \sigma_2 \sigma_1$. The identity permutation will be denoted e .

The **group algebra** of the symmetric group, denoted $\mathbb{C}[\mathbb{S}_n]$, is an $n!$ -dimensional complex¹ vector space with canonical basis $\{e_\sigma\}_{\sigma \in \mathbb{S}_n}$, equipped with a notion of vector/vector multiplication

¹Throughout the paper we use vector spaces defined over the complex numbers. This is because several of the general results from representation theory are much easier to state over \mathbb{C} than over \mathbb{R} . However, using the specific system of representations of \mathbb{S}_n called Young's Orthogonal Representation will allow us to perform all actual computations over the reals.

inherited from the structure of the symmetric group. In particular, $e_\sigma e_{\sigma'} = e_{\sigma\sigma'}$, and this is extended to the rest of the space by linearity. In other words, for general vectors $\mathbf{u} = \sum_{\sigma \in \mathbb{S}_n} u_\sigma e_\sigma$ and $\mathbf{v} = \sum_{\sigma \in \mathbb{S}_n} v_\sigma e_\sigma$,

$$\mathbf{u}\mathbf{v} = \sum_{\mu \in \mathbb{S}_n} \sum_{\nu \in \mathbb{S}_n} u_\mu v_\nu e_\mu e_\nu = \sum_{\mu, \nu \in \mathbb{S}_n} u_\mu v_\nu e_{\mu\nu} = \sum_{\sigma \in \mathbb{S}_n} w_\sigma e_\sigma, \quad (7)$$

where $w_\sigma = \sum_{\nu \in \mathbb{S}_n} u_{\sigma\nu^{-1}} v_\nu$. The inner product on $\mathbb{C}[\mathbb{S}_n]$ is the usual $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{\sigma \in \mathbb{S}_n} u_\sigma^* v_\sigma$, where $*$ stands for complex conjugation. For future reference we note that as a function w is called the **convolution** of v with u , and denoted $u \star v$.

Throughout the paper, if f is a function on permutations, then \mathbf{f} will be the corresponding group algebra vector $\sum_{\sigma \in \mathbb{S}_n} f(\sigma) e_\sigma$. Similarly, if $U \subset \mathbb{S}_n$, then \mathbf{U} will be $\sum_{\sigma \in U} e_\sigma$. In particular, we represent the total ranking (2) by the basis vector $\sigma \equiv e_\sigma$, and the partial rankings (3), (4) and (5) by $\mathbf{A}_{i_1, \dots, i_k} = \sum_{\sigma \in A_{i_1, \dots, i_k}} e_\sigma$, $\mathbf{B}_{i_1, \dots, i_k} = \sum_{\sigma \in B_{i_1, \dots, i_k}} e_\sigma$, and $\mathbf{C}_{i_1, 1 \dots i_\ell, k_\ell} = \sum_{\sigma \in C_{i_1, 1 \dots i_\ell, k_\ell}} e_\sigma$.

The significance of the multiplicative structure of $\mathbb{C}[\mathbb{S}_n]$ is that it allows us to express these partial ranking vectors as

$$\mathbf{A}_{i_1, \dots, i_k} = \Pi_k^n \mathbf{S}_{n-k} \pi_{i_1, \dots, i_k}, \quad (8)$$

$$\mathbf{B}_{i_1, \dots, i_k} = \mathbf{S}_{n-k} \pi_{i_1, \dots, i_k}, \quad (9)$$

$$\mathbf{C}_{i_1, 1 \dots i_\ell, k_\ell} = \Pi_{m_\ell}^n \dots \Pi_{m_1}^{m_2} \mathbf{S}_{n-k} \pi_{i_1, 1, \dots, i_\ell, k_1, \dots, i_\ell, k_\ell}, \quad (10)$$

where in the last equation $k = \sum_{i=1}^\ell k_i$, and π_{i_1, \dots, i_k} , Π_k^ℓ and \mathbf{S}_m are the following elementary sets of permutations.

- π_{i_1, \dots, i_k} is the **selector** permutation that maps i_1, \dots, i_k to positions $n, n-1, \dots, n-k+1$. Where it maps all other numbers is presently irrelevant, so we leave it undefined.
- Π_k^ℓ is the set of **interleavings** of $[n-\ell+1, n-k]$ with $[n-k+1, n]$, i.e., the set of permutations that satisfy

$$\begin{aligned} \sigma(i) &= i & \text{if } 1 \leq i \leq n-\ell, \\ \sigma(i) &< \sigma(j) & \text{if } n-\ell+1 \leq i < j \leq n-k, \\ \sigma(i) &< \sigma(j) & \text{if } n-k+1 \leq i < j \leq n. \end{aligned}$$

This concept is the same as the **riffle shuffles** introduced in (Huang & Guestrin, 2009).

- \mathbf{S}_m is the subgroup of permutations that only permute $[1, m]$ and leave $[m+1, n]$ fixed.

Intuitively, (8)–(10) describe the way that $\mathbf{A}_{i_1, \dots, i_k}$, $\mathbf{B}_{i_1, \dots, i_k}$ and $\mathbf{C}_{i_1, 1 \dots i_\ell, k_\ell}$ can be “built up” through a sequence of operations. For example, $\mathbf{A}_{i_1, \dots, i_k}$ is the set of permutations that we get by performing the following operations: (1) first map $i_1 \mapsto n$, $i_2 \mapsto n-1$, etc. up to $i_k \mapsto n-k+1$; (2) then permute all the “unoccupied” positions $[1, n-k]$ in all possible ways; (3) finally interleave the numbers occupying $[n-k+1, n]$ with whatever is in positions $[1, n-k]$, while preserving their relative ordering (see Figure 1). We will find that such factorizations are key to unraveling the spectral structure of partial rankings and for efficiently computing kernels. Before describing this, however, we need to develop some general results regarding kernels on permutations.

3 Kernels on the symmetric group

In studying kernels on \mathbb{R}^d a lot of attention is focused on translation invariance (or stationarity) in the sense of $k(x, x') = k(x+z, x'+z)$ for all $z \in \mathbb{R}^d$. When looking at kernels on non-commutative groups we find a similar notion of invariance, but now **right-invariance**, meaning $K(\sigma\pi, \sigma'\pi) = K(\sigma, \sigma')$ for all σ, σ' and π , and **left-invariance**, meaning $K(\pi\sigma, \pi\sigma') = K(\sigma, \sigma')$ for all σ, σ' and π , are distinct concepts.

Right-invariance is a natural requirement in ranking, since it captures the notion that if we take any permutation π , then the similarity between two rankings should not change if we relabel the underlying items x_1, \dots, x_n as $x_{\pi^{-1}(1)}, \dots, x_{\pi^{-1}(n)}$. A right-invariant kernel on \mathbb{S}_n can always be expressed as $K(\sigma', \sigma) = \kappa(\sigma'\sigma^{-1})$ for some $\kappa: \mathbb{S}_n \rightarrow \mathbb{R}$. A function $\kappa: \mathbb{S}_n \rightarrow \mathbb{C}$ is said to be a **positive (semi-)definite function** on \mathbb{S}_n if the corresponding $K(\sigma', \sigma) = \kappa(\sigma'\sigma^{-1})$ is a positive (semi-)definite kernel. Note that in this case $\kappa(\sigma) = K(\sigma, e) = K(e, \sigma) = \kappa(\sigma^{-1})$.

Left-invariance is not desirable in ranking, since it would imply that the value of the kernel between a pair of rankings that rank a specific item in positions one and two respectively should be the same as if they ranked it in positions, say, one and twenty (assuming that the ranking of

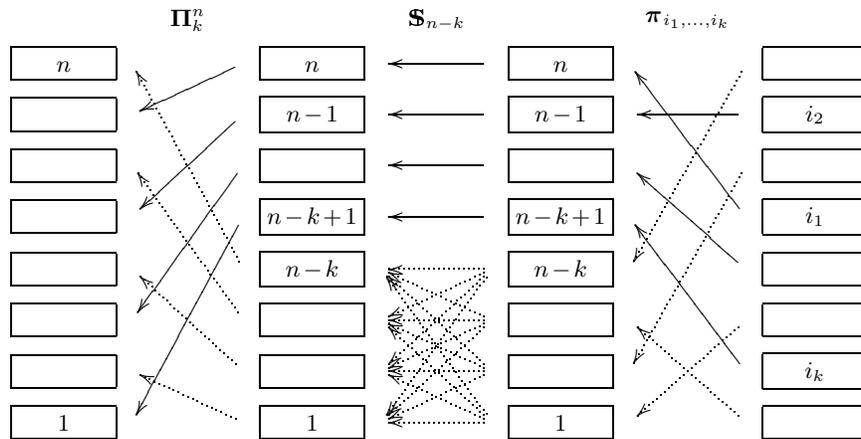


Figure 1: An illustration of the factorization $\mathbf{A}_{i_1, \dots, i_k} = \Pi_k^n \mathbf{S}_{n-k} \pi_{i_1, \dots, i_k}$. First, π_{i_1, \dots, i_k} moves (i_1, i_2, \dots, i_k) into positions $(n, n-1, \dots, n-k+1)$. Next, \mathbf{S}_{n-k} permutes positions $1, 2, \dots, n-k$ in all possible ways. Finally, Π_k^n maps $n, n-1, \dots, n-k+1$ to $1, 2, \dots, n$ in all possible ways that respects their relative ordering.

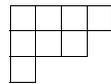
other shared items does not change). However, in permutation learning, for example when we are trying to learn the optimal assignment between aircraft and routes, left-invariance might be just as natural a requirement as right-invariance. A kernel which is both left- and right-invariant is called **bi-invariant**. In the bi-invariant case κ is a class function, which means that $\kappa(\tau^{-1}\sigma\tau) = \kappa(\sigma)$ for all τ and σ .

On \mathbb{R}^d Bochner's theorem tells us that positive definite functions are characterized by the fact that their Fourier transform is pointwise positive. On finite non-commutative groups we have a similar result, except that now the ordinary **Fourier transform** must be replaced by its non-commutative generalization. In the case of the symmetric group this takes the form

$$\widehat{\kappa}(\lambda) = \sum_{\sigma \in \mathbb{S}_n} \kappa(\sigma) \rho_\lambda(\sigma) \quad \lambda \vdash n, \quad (11)$$

and differs from the usual commutative Fourier transforms in two key respects:

1. Instead of frequency, the individual Fourier components are now indexed by **integer partitions** of n , by which we mean a sequence $\lambda = (\lambda_1, \dots, \lambda_k)$ of weakly decreasing positive integers summing to n (i.e., $\lambda_1, \dots, \lambda_k \in \mathbb{Z}$, $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k$, $\sum_{i=1}^k \lambda_i = n$). A convenient graphical representation for partitions is provided by so-called **Young diagrams**, consisting of $\lambda_1, \dots, \lambda_k$ boxes placed in consecutive rows. For example,



is the Young diagram of $\lambda = (4, 3, 1)$. The notation $\lambda \vdash n$ just means that λ is an integer partition of n .

2. Instead of expressions like $e^{ikx/2\pi}$, the weighting factors $\rho_\lambda(\sigma)$ appearing in (11) are complex-valued *matrices*, specifically elements of the **irreducible representation** (or **irrep**) of \mathbb{S}_n corresponding to λ . For the definition of what this means and how to construct such matrices the reader is referred to the supplementary document. For now what is important to note is that the Fourier transform $(\widehat{\kappa}(\lambda))_{\lambda \vdash n}$ is a sequence of matrices of different sizes. We use d_λ to denote the dimensionality of the irrep indexed by λ , hence $\widehat{\kappa}(\lambda) \in \mathbb{C}^{d_\lambda \times d_\lambda}$. We also assume throughout the paper that the irreps are unitary, and hence $\rho_\lambda(\sigma^{-1}) = \rho_\lambda(\sigma)^\dagger$.

Aside from these two perhaps surprising features, (11) shares many important properties with ordinary Fourier transformation. In particular, with respect to the appropriate matrix norms, $\kappa \mapsto \widehat{\kappa}$ is a unitary transformation, the inverse transform being

$$\kappa(\sigma) = \frac{1}{n!} \sum_{\lambda \vdash n} d_\lambda \operatorname{tr}[\widehat{\kappa}(\lambda) \rho_\lambda(\sigma^{-1})],$$

and we also have analogs of the convolution and correlation theorems. By unitarity, the total size of the Fourier matrices must be the same as the size of the original function, that is, $\sum_{\lambda \vdash n} d_\lambda^2 = n!$. Bochner’s theorem generalizes as follows. We do not claim that this result is novel to mathematics, so the attributions only serve to indicate its first appearance in the machine learning literature.

Proposition 1 (Kondor, 2008, Thm. 4.5.4)(Fukumizu et al., 2009, Thm. 11) *A function $\kappa: \mathbb{S}_n \rightarrow \mathbb{C}$ is positive (semi-)definite if and only if each of the $\widehat{\kappa}(\lambda)$ matrices in (11) are positive (semi-)definite.*

Much of the following discussion will revolve around moving back and forth between three different views of the same objects: (1) the function view ($f: \mathbb{S}_n \rightarrow \mathbb{C}$); (2) the group algebra vector view ($\mathbf{f} \in \mathbb{C}[\mathbb{S}_n]$); and (3) the Fourier transform view $\widehat{\mathbf{f}}$. Therefore, we will use the symbol $\widehat{}$ very liberally: if it is placed over a group algebra element, it will denote the Fourier transform of the corresponding function, and if it is placed over a group element or set of group elements, it will denote the Fourier transform of the corresponding indicator function.

3.1 Diffusion kernels

While Proposition 1 gives a general characterization of right-invariant kernels on the symmetric group, it does not give us much guidance as to what specific kernel we should use for ranking. One way to derive a kernel would be to induce it from a right-invariant metric on \mathbb{S}_n , for example, one of the metrics described in (Diaconis, 1988). However, if we then wanted to perform operations on the kernel in Fourier space, as we will do in the next section, we would at some point have to perform an explicit Fourier transform, which is very expensive.²

To avoid this problem, in the present paper we use **diffusion kernels** (Kondor & Lafferty, 2002), which have strong connections to spectral theory. Recall that to define a diffusion kernel we start with an adjacency relation \sim , which induces a graph. The corresponding **graph Laplacian** is the matrix

$$\Delta_{\sigma',\sigma} = \begin{cases} 1 & \text{if } \sigma' \sim \sigma \\ -d_\sigma & \text{if } \sigma' = \sigma \\ 0 & \text{otherwise,} \end{cases}$$

and the diffusion kernel is $K(\sigma',\sigma) = [e^{\beta\Delta}]_{\sigma',\sigma}$ for some diffusion parameter $\beta \in \mathbb{R}$, where $e^{\beta\Delta}$ is the matrix exponential $\lim_{m \rightarrow \infty} (I + \beta\Delta/m)^m$.

A diffusion kernel on a group is right-invariant if and only if \sim is a right-invariant relation, which is equivalent to saying that $\sigma' \sim \sigma \iff \sigma'\sigma^{-1} \in Q$ for some $Q \subset \mathbb{S}_n$. To ensure symmetry, Q must also be symmetric in the sense that $\pi \in Q \iff \pi^{-1} \in Q$. The set Q can be interpreted as the “elementary steps” $\sigma \mapsto \pi\sigma$ that one can take from any σ to reach its neighbors. In fact, for the graph to be connected, Q must be a generating set of \mathbb{S}_n . In this case the adjacency graph is known as the **Cayley graph** induced by Q .

The natural generalization of the above to weighted graphs involves setting $\Delta_{\sigma',\sigma} = q(\sigma'\sigma^{-1})$, where q is a function that must satisfy $q(\pi^{-1}) = q(\pi)$ and $\sum_{\pi \in \mathbb{S}_n} q(\pi) = 0$. The following result characterizes all possible right-invariant diffusion kernels on \mathbb{S}_n and shows how to compute them in Fourier space.

Proposition 2 *If $\Delta_{\sigma',\sigma} = q(\sigma'\sigma^{-1})$ and q satisfies the above two conditions, then the diffusion kernel $K(\sigma',\sigma) = [e^{\beta\Delta}]_{\sigma',\sigma}$ is right-invariant, and $K(\sigma',\sigma) = \kappa(\sigma'\sigma^{-1})$, where $\widehat{\kappa}(\lambda) = \exp(\beta\widehat{q}(\lambda))$ for each $\lambda \vdash n$.*

Proposition 2 tells us that instead of exponentiating the $n!$ -dimensional matrix Δ (at a cost of $O(n!^3)$), it is more efficient to compute the diffusion kernel in Fourier space, where we only have to exponentiate individual Fourier matrices (at a total cost of $O(\sum_{\lambda \vdash n} d_\lambda^3)$). The question remains as to what adjacency relation \sim is appropriate for ranking problems.

A **transposition** (i, j) is a permutation that swaps i with j and leaves everything else fixed. Since transpositions are in many ways the simplest non-trivial permutations, they seem like a natural choice for Q . However, because transpositions form a **conjugacy class** (i.e., $\pi \in Q \implies \mu^{-1}\pi\mu \in$

²Naively, the complexity of computing a Fourier transform on \mathbb{S}_n is $O(n!^2)$. Fast Fourier transforms, such as Clausen’s FFT (Clausen, 1989), can bring this complexity down to $O(n^3n!)$ or even $O(n^2n!)$ (Maslen, 1998), but the the $n!$ factor still makes using such kernels infeasible for large n , unless we can derive the form of their Fourier transform analytically.

$Q \forall \mu$), the resulting K_{transp} will be bi-invariant, so, for the reasons we discussed above, this type of kernel is more appropriate for permutation learning than for ranking. From a purely algebraic point of view, however, K_{transp} is a canonical choice, showing some similarities with the Gaussian RBF kernel on \mathbb{R}^d . In fact, using a result on page 40 of (Diaconis, 1988), the Fourier transform of this kernel can be derived in closed form.

Proposition 3 *If $K_{\text{transp}}: \mathbb{S}_n \times \mathbb{S}_n \rightarrow \mathbb{R}$ is the diffusion kernel induced by the class of transpositions, then*

$$\widehat{\kappa}_{\text{transp}}(\lambda) = \exp(-\beta \binom{n}{2} (1 - r(\lambda))) I_{d_\lambda}, \quad (12)$$

where $r(\lambda) = \binom{n}{2}^{-1} \sum_i \binom{\lambda_i}{2} - \binom{\lambda'_i}{2}$, λ' is the transpose of the partition λ , and I_{d_λ} denotes the d_λ -dimensional identity matrix.

For ranking a better choice of kernel is K_{adj} , the diffusion kernel induced by the subset of **adjacent transpositions** $\{\tau_i := (i, i+1)\}_{i=1}^{n-1}$. This is the kernel that we use in our experiments. Unfortunately, we have no closed form expression for $\widehat{\kappa}_{\text{adj}}$. While \widehat{q}_{adj} is relatively easy to compute by a direct Fourier transform since there are only $n-1$ adjacent transpositions, for large n exponentiating these matrices might be problematic. Note, however, that this is a one-time computation.

A natural variant on K_{adj} is to give different adjacent transpositions different weights. In learning from the top- k rankings returned by search engines, for example, we could give less weight to adjacent transpositions between the first few rankings than those lower down, reflecting the fact that whether something is ranked first or second is much more important than whether it is ranked, say 15th or 16th. While we do not pursue this direction further in the rest of the paper, we note that our computational results would hold for this variant of K_{adj} equally well.

4 Computing kernels in Fourier space

We started our discussion by stating that in typical ranking problems individual examples are not total rankings, but partial rankings of k out of n items. The easiest and most general way to extend the kernels of the previous section to this setting is to define the kernel between a pair of partial rankings R and R' as

$$K(R', R) = \frac{1}{|R'| |R|} \sum_{\sigma' \in R'} \sum_{\sigma \in R} K(\sigma', \sigma), \quad (13)$$

where, overloading notation somewhat, R and R' double as the set of all permutations satisfying the two partial rankings.

The advantage of such an averaged kernel is its flexibility: R and R' can be any type of partial rankings (interleaving, top- k , or some multiranking combining elements of the former), and even the orders (k and k') of R and R' may be different. This allows kernel-based ranking algorithms to aggregate information from a diverse array of inputs.

The problem with (13) is that it appears to be very expensive to compute: naively, its complexity is $O((n-k)!(n-k')!)$. The rest of the paper addresses this computational issue, showing that (13) can be computed efficiently in Fourier space. We begin with the following two general lemmas that both follow from the convolution theorem, which states that $\widehat{u \star v}(\lambda) = \widehat{u}(\lambda) \widehat{v}(\lambda)$.

Lemma 4 *If $\mathbf{u}, \mathbf{v} \in \mathbb{C}[\mathbb{S}_n]$ and $\mathbf{w} = \mathbf{u}\mathbf{v}$, then $\widehat{w}(\lambda) = \widehat{u}(\lambda) \widehat{v}(\lambda)$ for each $\lambda \vdash n$.*

Lemma 5 *For any $\mathbf{u}, \mathbf{v} \in \mathbb{C}[\mathbb{S}_n]$,*

$$\langle \mathbf{u}, \mathbf{v} \rangle = \frac{1}{n!} \sum_{\lambda \vdash n} d_\lambda \text{tr}[\widehat{u}(\lambda)^\dagger \widehat{v}(\lambda)], \quad (14)$$

where † denotes the Hermitian conjugate.

Using these lemmas it is easy to derive the Fourier form of our kernel.

Proposition 6 *If K is right-invariant with $K(\sigma', \sigma) = \kappa(\sigma' \sigma^{-1})$, then (13) can be expressed as*

$$K(R', R) = \frac{1}{n! |R'| |R|} \sum_{\lambda \vdash n} d_\lambda \text{tr}[\widehat{R}'(\lambda)^\dagger \widehat{\kappa}(\lambda) \widehat{R}(\lambda)]. \quad (15)$$

Proof. By right-invariance

$$\sum_{\sigma' \in R'} \sum_{\sigma \in R} K(\sigma', \sigma) = \sum_{\sigma' \in \mathbb{S}_n} \sum_{\sigma \in \mathbb{S}_n} R'(\sigma') K(\sigma', \sigma) R(\sigma) = \sum_{\sigma' \in \mathbb{S}_n} \sum_{\sigma \in \mathbb{S}_n} R'(\sigma') \kappa(\sigma', \sigma^{-1}) R(\sigma) = \langle \mathbf{R}', \boldsymbol{\kappa} \mathbf{R} \rangle,$$

where, as before, $R(\sigma)$ is the indicator function of the set R and $\mathbf{R} = \sum_{\sigma \in R} \mathbf{e}_\sigma$ is the corresponding vector in the group algebra. Hence,

$$K(R', R) = |R'|^{-1} |R|^{-1} \langle \mathbf{R}', \boldsymbol{\kappa} \mathbf{R} \rangle, \quad (16)$$

and (15) follows by Lemmas 4 and 5. ■

By itself, (15) does not make our kernel any easier to evaluate, since the combined size of the matrices appearing under the sum is still $O(n!)$. The key is to additionally exploit the sparsity of \widehat{R} and \widehat{R}' . We derive the structure of these Fourier transforms in two stages: first describing their matrix-level sparsity, and then examining the row/column-level sparsity of their individual matrix components.

4.1 Matrix level sparsity

We say that a vector $\mathbf{v} \in \mathbb{C}[\mathbb{S}_n]$ (equivalently, the function v or the Fourier transform \widehat{v}) is **bandlimited** to some subset Λ of $\{\lambda \vdash n\}$ if the only non-zero components of \widehat{v} are $\{\widehat{v}(\lambda)\}_{\lambda \in \Lambda}$. Several recent papers have used bandlimited functions to approximate distributions over permutations, most notably in the context of multi-object tracking (Kondor et al., 2007; Huang et al., 2009; Huang & Guestrin, 2009).

In the present paper bandlimitedness is not an approximation, but an inherent feature of our problem. Typically, this is a sign of invariance to a subgroup, in our case, invariance to the ranking position of the $n-k$ items not involved in the partial ranking at hand. The key to unraveling this structure are the factorizations (8)–(10), and specifically, the \mathbf{S}_{n-k} factors appearing in them.

Proposition 7 *The group algebra element $\mathbf{S}_{n-k} \in \mathbb{C}[\mathbb{S}_n]$ is bandlimited to the set Λ_{n-k}^n defined $\Lambda_{n-k}^n = \{\lambda \vdash n \mid \lambda_1 \geq n-k\}$.*

In terms of Young diagrams, Λ_{n-k}^n is the set of diagrams of n boxes with at least $n-k$ boxes in their first row. For example, $\Lambda_7^{10} = \{(10), (9, 1), (8, 2), (8, 1, 1), (7, 3), (7, 2, 1), (7, 1, 1, 1)\}$, so the Fourier transform of \mathbf{S}_7 in $\mathbb{C}[\mathbb{S}_{10}]$ has just 7 non-zero components. An immediate consequence of Lemma 4 is that if \mathbf{u} is Λ_u -bandlimited and \mathbf{v} is Λ_v -bandlimited, then \mathbf{uv} will be $\Lambda_u \cap \Lambda_v$ -bandlimited. Thus, the k 'th order partial ranking vectors (8)–(10) will all inherit the bandlimited structure of \mathbf{S}_{n-k} , and the summation in (15) need only extend over $\Lambda_{n-\min(k,k')}^n$, giving us the following result.

Proposition 8 *If R and R' are a pair of partial rankings of orders k and k' , respectively, then the kernel (15) can be written as*

$$K(R', R) = \frac{1}{n! |R'| |R|} \sum_{\lambda \in \Lambda_{n-\min(k,k')}^n} d_\lambda \operatorname{tr} [\widehat{R}'(\lambda)^\dagger \widehat{\kappa}(\lambda) \widehat{R}(\lambda)]. \quad (17)$$

In particular, given \widehat{R} , \widehat{R}' and $\widehat{\kappa}$, the kernel can be computed in $\sum_{\lambda \in \Lambda_{n-\min(k,k')}^n} 2d_\lambda^3$ operations.³

Several authors (Diaconis, 1988; Huang et al., 2009; Kondor et al., 2007) discuss that one possible interpretation of the Fourier matrices is that they capture detail at different scales: given a distribution p on \mathbb{S}_n , $(\widehat{p}(\lambda))_{\lambda \in \Lambda_{n-k}^n}$ is exactly the information needed to reconstruct p up to its k 'th order marginals. In this respect it is not surprising that (17) should involve exactly these Fourier matrices, and our result fits nicely in the general theory of spectral analysis on permutations.

Unfortunately, in general, the dimensionality of the largest matrices in $(R(\lambda))_{\lambda \in \Lambda_{n-k}^n}$ grows with $O(n^k)$, so while Proposition 8 greatly reduces the number of Fourier matrices that need to be summed over, for n greater than about a dozen evaluating (17) is still problematic (see Table 1). This motivates a finer grained analysis, also taking the internal structure of the Fourier matrices into account.

³Throughout this paper, in line with the literature, by a single operation we mean multiplying two scalars and adding them to a third. We assume that multiplication by constants and copying information is free.

$$\widehat{\mathbf{S}}_3((5)) = (20) \quad \widehat{\mathbf{S}}_3((4, 1)) = \begin{pmatrix} 20 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\widehat{\mathbf{S}}_3((3, 2)) = \begin{pmatrix} 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \widehat{\mathbf{S}}_3((3, 1, 1)) = \begin{pmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 2: The non-zero Fourier matrices of $\mathbf{S}_3 \in \mathbb{C}[\mathbb{S}_5]$, relevant to any binary ranking out of 5 items. Any $\mathbf{S}_{n-2} \in \mathbb{C}[\mathbb{S}_n]$ would have the same structure except that for $n > 5$ the matrices would be larger.

4.2 Row/column level sparsity

Ultimately, the sparsity of $\widehat{\mathbf{S}}_{n-k}$ is a consequence of the way that the irreps of \mathbb{S}_n reduce into a direct sum of irreps of \mathbb{S}_{n-k} on restriction to this subgroup. Specifically, a result called **Young's branching rule** tells us that if $\tau \in \mathbb{S}_{n-1}$, then

$$\rho_\lambda(\tau) = T^{-1} \left[\bigoplus_{\lambda^- \in \lambda \downarrow_{n-1}} \rho_{\lambda^-}(\tau) \right] T,$$

where $\lambda \downarrow_{n-1}$ is the set of all partitions of $n-1$ that we can get from λ by removing a single box, ρ_{λ^-} is the irrep of \mathbb{S}_{n-1} indexed by λ^- , and T is a unitary matrix that depends on exactly what system of irreps we use. For simplicity, in the following we will assume that all irreps are from **Young's Orthogonal Representation (YOR)** (see supplement), in which case the T matrices may be dropped because they are always equal to the identity.

Now if we establish a partial order on partitions according to which $\lambda \geq \lambda'$ if and only if λ' is a subdiagram of λ , recursively applying the branching rule down to $n-k$ gives that for $\lambda' \vdash n-k$ and $\tau \in \mathbb{S}_{n-k}$, $\rho_{\lambda'}(\tau)$ will be featured in the decomposition of $\rho_\lambda(\tau)$ if and only if $\lambda \geq \lambda'$. In particular, $\rho_{(n-k)}(\tau)$ appears whenever $\lambda \geq (n-k)$, which is equivalent to $\lambda \in \Lambda_{n-k}^n$. Proposition 7 follows by considering that $\sum_{\tau \in \mathbb{S}_{n-m}} \rho_{\lambda'}(\tau) = 0$ for all $\lambda' \vdash n-k$ except for $\lambda' = (n-k)$ (see the proof of Proposition 9 in the supplement).

The precise structure of $\widehat{\mathbf{S}}_{n-k}$ can be uncovered by repeating this analysis on the level of individual matrix entries. First recall that the individual rows/columns of Fourier matrices such as $\widehat{\mathbf{S}}_{n-k}(\lambda)$ are indexed by the **standard Young tableaux (SYT)** of shape λ , such as

$$t = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 4 & 5 & 7 \\ \hline 3 & 6 & 9 & & \\ \hline 8 & & & & \\ \hline \end{array},$$

which is a SYT of shape $\lambda = (5, 3, 1)$. Just like partitions, SYT also have a natural inclusion order, for example if $t' = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 4 & 5 & 7 \\ \hline 3 & 6 & & & \\ \hline & & & & \\ \hline \end{array}$, then $t \geq t'$ because t can be constructed from t' by adding $\boxed{8}$ and $\boxed{9}$. A convenient shorthand for SYT are Yamanouchi symbols, in particular, $[\dots]_m$ denotes $\begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & \dots & m \\ \hline \end{array}$. Using these notations, the branching rule in YOR can be made more explicit as

$$[\rho_\lambda(\tau)]_{t,t'} = \begin{cases} [\rho_{\lambda^-}(\tau)]_{u,u'} & \text{if } u \leq t, u' \leq t', \text{ and } u \text{ and } u' \text{ are both of shape } \lambda^- \vdash n-1, \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

Using this result we can derive the exact form of $\widehat{\mathbf{S}}_{n-k}$.

Proposition 9 For $0 \leq k \leq n-1$ the Fourier transform of $\mathbf{S}_{n-k} \in \mathbb{C}[\mathbb{S}_n]$ (in YOR) is of the form

$$[\widehat{\mathbf{S}}_{n-k}(\lambda)]_{t,t'} = \begin{cases} (n-k)! & \text{if } t = t' \text{ and } t \geq [\dots]_{n-k}, \\ 0 & \text{otherwise.} \end{cases}$$

In simple terms $t \geq [\dots]_{n-k}$ means that the first $m = n - k$ boxes in the first row of t must be $\overline{[1|2|3|\dots|m]}$, which is a very severe restriction. Thus, not only is \mathbb{S}_{n-k} bandlimited to just a few matrix components, even those components will have few non-zero entries. As an example, Figure 2 shows the actual Fourier matrices of $\mathbb{S}_3 \in \mathbb{C}[\mathbb{S}_5]$. Taking advantage of this form of sparsity is more difficult than just taking advantage of bandlimitedness because in expressions such as (8) \mathbb{S}_{n-k} is multiplied from both the left and the right. To overcome this problem, we need to introduce adjoints.

Proposition 10 *For any $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{C}[\mathbb{S}_n]$, there is a $\mathbf{v}^\dagger \in \mathbb{C}[\mathbb{S}_n]$ called the **adjoint** of \mathbf{v} , such that*

$$\langle \mathbf{u}, \mathbf{v}\mathbf{w} \rangle = \langle \mathbf{v}^\dagger \mathbf{u}, \mathbf{w} \rangle \quad \text{and} \quad \langle \mathbf{u}, \mathbf{w}\mathbf{v} \rangle = \langle \mathbf{u}\mathbf{v}^\dagger, \mathbf{w} \rangle.$$

Moreover, $\mathbf{v}^\dagger(\sigma) = v(\sigma^{-1})^*$ for all $\sigma \in \mathbb{S}_n$, and in YOR $(\widehat{\mathbf{v}^\dagger})(\lambda) = \widehat{\mathbf{v}}(\lambda)^\dagger$ for all $\lambda \vdash n$.

Substituting a pair of interleaving rankings $\mathbf{A}_{i_1, \dots, i_k}$ and $\mathbf{A}_{i'_1, \dots, i'_k}$ for \mathbf{R} and \mathbf{R}' in (16) we can now rearrange the inner product as

$$\begin{aligned} & \langle \mathbf{A}_{i'_1, \dots, i'_k}, \boldsymbol{\kappa} \mathbf{A}_{i_1, \dots, i_k} \rangle = \\ & \langle \mathbf{\Pi}_k^n \mathbf{S}_{n-k} \boldsymbol{\pi}_{i'_1, \dots, i'_k}, \boldsymbol{\kappa} \mathbf{\Pi}_k^n \mathbf{S}_{n-k} \boldsymbol{\pi}_{i_1, \dots, i_k} \rangle = \\ & \langle \mathbf{S}_{n-k} \boldsymbol{\pi}_{i'_1, \dots, i'_k} \boldsymbol{\pi}_{i_1, \dots, i_k}^\dagger \mathbf{S}_{n-k}^\dagger, \mathbf{\Pi}_k^{n\dagger} \boldsymbol{\kappa} \mathbf{\Pi}_k^n \rangle = \\ & \langle \mathbf{S}_{n-k} \boldsymbol{\pi}_{i'_1, \dots, i'_k} \boldsymbol{\pi}_{i_1, \dots, i_k}^{-1} \mathbf{S}_{n-k}, \mathbf{\Pi}_k^{n\dagger} \boldsymbol{\kappa} \mathbf{\Pi}_k^n \rangle, \end{aligned}$$

where the last line follows from $\mathbf{S}_{n-k}^\dagger = \mathbf{S}_{n-k}$ and $\boldsymbol{\pi}_{i_1, \dots, i_k}^\dagger = \boldsymbol{\pi}_{i_1, \dots, i_k}^{-1}$. Now the first argument of the last inner product contains an expression sandwiched between two \mathbf{S}_{n-k} 's, which has the effect of zeroing out all rows and columns indexed by $t \not\geq [\dots]_{n-k}$. This dramatically reduces the effective size of the Fourier matrices that we need to multiply together to compute the kernel. Introducing the notation $[M]_{\geq [\dots]_{n-k}}$ for the submatrix of the Fourier matrix indexed by rows/columns indexed by SYT descended from $[\dots]_{n-k}$, we have the following result.

Proposition 11 *If $R = \mathbf{A}_{i_1, i_2, \dots, i_k}$ and $R' = \mathbf{A}_{i'_1, i'_2, \dots, i'_k}$, then the kernel (13) can be expressed as*

$$K(R, R') = \frac{(n-k)!^2}{n! |R'| |R|} \sum_{\lambda \in \Lambda_{n-k}^n} d_\lambda [\widehat{\Omega}(\lambda)]_{\geq [\dots]_{n-k}} \odot [\widehat{\bar{\kappa}}(\lambda)]_{\geq [\dots]_{n-k}}, \quad (19)$$

where $\Omega = \boldsymbol{\pi}_{i'_1, \dots, i'_k} \boldsymbol{\pi}_{i_1, \dots, i_k}^{-1}$, $\bar{\kappa} = \mathbf{\Pi}_k^{n\dagger} \boldsymbol{\kappa} \mathbf{\Pi}_k^n$ and \odot denotes the matrix inner product $A \odot B = \sum_{i,j} A_{i,j} B_{i,j}$.

The restricted matrices appearing in (19) are much smaller than the matrices that we had to multiply together in (15) (see Table 1), and what is particularly attractive is that their size is *independent of* n . To be fair, (19) also requires computing $[\widehat{\Omega}(\lambda)]_{\geq [\dots]_{n-k}}$. The next section explains how to do that efficiently, with a complexity that does not grow with n , either.

Clearly, formulae similar to (19) also hold for the more general case $k \neq k'$, as well as for other types of partial rankings. For example, if R and R' are both top- k rankings, then all that we need to change is to set $\bar{\kappa} = \boldsymbol{\kappa}$. If R is an interleaving ranking, but R' is a top- k ranking, then $\bar{\kappa} = \boldsymbol{\kappa} \mathbf{\Pi}_k^n$, and so on.

5 Complexity

A function $f: \mathbb{S}_n \rightarrow \mathbb{C}$ is called **right \mathbb{S}_{n-k} -invariant** if $f(\sigma\tau) = f(\sigma)$ for all $\tau \in \mathbb{S}_{n-k}$. Clearly, the space spanned by these functions has dimension $n!/(n-k)!$. In (Kondor et al., 2009) it was argued that such functions are bandlimited to $\{\widehat{f}(\lambda)\}_{\lambda \in \Lambda_{n-k}^n}$ and that their Fourier transforms fully occupy at least one column in each of these matrices. Therefore, $\sum_{\lambda \in \Lambda_{n-k}^n} d_\lambda \leq n!/(n-k)!$, and thus, even if we assume that $\widehat{\kappa}(\lambda)$, $\widehat{R}(\lambda)$ and $\widehat{R}'(\lambda)$ have all been pre-computed, the complexity of computing (17) is

$$\sum_{\lambda \in \Lambda_{n-k}^n} 2d_\lambda^3 \leq 2(n!/(n-k)!)^3 = O(n^{3k})$$

for fixed $k = k'$. See Table 1 for the exact operation count for $n = 20$ and some small values of k .

In contrast, denoting the set of all SYT of shape λ by \mathcal{T}^λ , and denoting its subset of SYT descended from $[\dots]_{n-k}$ by $\mathcal{T}_{n-k}^\lambda$, computing (19) only requires $\sum_{\lambda \in \Lambda_{n-k}^n} (|\mathcal{T}_{n-k}^\lambda|)^2$ operations. Now

k	2	3	4	5	6	7
c_0	$2.0 \cdot 10^7$	$1.8 \cdot 10^{10}$	$7.0 \cdot 10^{12}$	$4.2 \cdot 10^{15}$	$2.6 \cdot 10^{17}$	$2.6 \cdot 10^{19}$
c_1	7	34	209	1,546	13,327	130,922
c_2	1068	$8,7 \cdot 10^4$	$7.7 \cdot 10^6$	$7.9 \cdot 10^8$	$9.2 \cdot 10^{10}$	$1.2 \cdot 10^{13}$
$(2k)^{2k+2}$	4096	$1.7 \cdot 10^6$	$1.0 \cdot 10^9$	$1.0 \cdot 10^{12}$	$1.3 \cdot 10^{15}$	$2.2 \cdot 10^{18}$

Table 1: Comparison of the cost of computing the kernel with different methods. Here c_0 is the cost of computing (17) using naive matrix multiplication for $n = 20$. In contrast, c_1 is the cost of computing (19) given the $\widehat{\Omega}(\lambda)$ and $\widehat{\kappa}(\lambda)$ Fourier matrices (irrespective of n). The maximum number of operations required to compute $[\widehat{\Omega}(\lambda)]_{\geq[\dots]_{n-k}}$ as an input to (19) is c_2 . Finally, the last row is our (loose) upper bound on c_2 .

assuming that $n \geq 2k$, imagine that we construct each $t \in \bigcup_{\lambda \in \Lambda_{n-k}^n} \mathcal{T}_{n-k}^\lambda$ in two stages: first deciding which of the numbers $[n-k+1, n]$ should appear in row one, and then placing the remaining, say ℓ , numbers in subsequent rows. As a result of this placement, rows two and higher will form a diagram λ' of their own, and the number of ways that λ' can be filled with the ℓ numbers is the same as the number of ways that it could be filled with $[1, \ell]$, i.e., $|\mathcal{T}^{\lambda'}|$. In summary, the total number of entries in the matrices appearing in (19) is

$$\sum_{\ell=0}^k \binom{k}{\ell}^2 \sum_{\lambda \vdash \ell} |\mathcal{T}^{\lambda'}|^2. \quad (20)$$

By the unitarity of the Fourier transform, we know that the total size of the Fourier matrices must be the same as the size of the group, so $\sum_{\lambda \vdash \ell} |\mathcal{T}^{\lambda'}|^2 = \ell!$, giving a complexity of $\sum_{\ell=0}^k \binom{k}{\ell}^2 \ell! = O(k^{2k+1})$ for computing all the matrix inner products in (19) (see “ c_1 ” in Table 1).

It remains to quantify the complexity of computing $[\widehat{\Omega}(\lambda)]_{\geq[\dots]_{n-k}}$. This hinges on the special structure of YOR, namely that if τ is an adjacent transposition, then $\rho_\lambda(\tau)$ has only two non-zero elements in each row (or column) and therefore for any $M \in \mathbb{C}^{d_\lambda \times d_\lambda}$ the product $\rho_\lambda(\tau)M$ takes only $2d_\lambda^2$ operations to compute. By a bubblesort type procedure (see, e.g., (Kondor et al., 2009) or the original \mathbb{S}_n FFT paper (Clausen, 1989)) any permutation can be decomposed into a product of at most $\binom{n}{2}$ transpositions, so if $\sigma = \pi_{i'_1, \dots, i'_k} \pi_{i_1, \dots, i_k}^{-1}$, then $\widehat{\sigma}(\lambda)$ can be computed in $n(n-1)d_\lambda^2$ operations. Since, as we have seen, $d_\lambda = O(n^k)$, computing $\widehat{\Omega}(\lambda)$ this way would make the total complexity of kernel evaluations $O(n^{2k+2})$, which is not much better than what we had for (15).

5.1 Relabeling

To address this problem we observe that the role of $\pi_{i_1, \dots, i_k}^{-1}$ is to map $[n-k+1, n]$ to some item labels, and $\pi_{i'_1, \dots, i'_k}$ maps some of those labels back to $[n-k+1, n]$. However, when we sandwich this product between two \mathbf{S}_{n-k} 's, where exactly we map $[1, n-k]$, and what is mapped to $[1, n-k]$ does not matter. This lets us “relabel” our items so that the permutations only touch $[n-2k+1, n]$. More formally, we can find a pair of permutations μ and μ' that fix $[1, n-2k]$ and have the property that

$$\mathbf{S}_{n-k} \pi_{i'_1, \dots, i'_k} \pi_{i_1, \dots, i_k}^{-1} \mathbf{S}_{n-k} = \mathbf{S}_{n-k} \mu' \mu^{-1} \mathbf{S}_{n-k}. \quad (21)$$

Once again, we find that in YOR the representation matrices of such permutations have a special form. If $T \subset \mathcal{T}^\lambda$, then we say that $[\rho_\lambda(\sigma)]_{T, T}$ is a **block** in $\rho_\lambda(\sigma)$ if $[\rho_\lambda(\sigma)]_{t, t'} = 0$ whenever $t \in T$, but $t' \notin T$ or vice versa. By the definition of YOR, if τ_i is the adjacent transposition $(i, i+1)$ and $[\rho_\lambda(\tau_i)]_{t, t'} \neq 0$, then t and t' must differ by at most the position of i and i' in their tableaux. In particular, if $t \in \mathcal{T}_m^\lambda$, then the first row of t starts with boxes numbered $1, \dots, m$, so if $i > m$ and $[\rho_\lambda(\tau_i)]_{t, t'} \neq 0$, then the first row of t' must also start with $1, \dots, m$, i.e., $[\rho_\lambda(\tau_i)]_{\mathcal{T}_m^\lambda, \mathcal{T}_m^\lambda}$ is a block. Since any σ that fixes $[1, m]$ can be written as a product of such adjacent transpositions, we have the following result.

Proposition 12 *If $\sigma \in \mathbb{S}_n$ fixes $[1, m]$, then in YOR $[\rho_\lambda(\sigma)]_{\mathcal{T}_m^\lambda, \mathcal{T}_m^\lambda}$ is a block in $\rho_\lambda(\sigma)$ for any $\lambda \in \Lambda_m^n$.*

Letting $\sigma = \mu' \mu^{-1}$ from (21), Proposition 12 tells us that $[\rho_\lambda(\sigma)]_{\mathcal{T}_{n-2k}^\lambda, \mathcal{T}_{n-2k}^\lambda}$ is a block, and clearly $[\widehat{\Omega}(\lambda)]_{\geq[\dots]_{n-k}}$ needed by (19) is a submatrix of this block. Therefore, to compute $[\widehat{\Omega}(\lambda)]_{\geq[\dots]_{n-k}}$ we need only construct this block and not the whole matrix $\widehat{\Omega}(\lambda)$. Using the same argument as what lead to (20) and multiplying by the $\binom{2k}{2}$ adjacent transpositions necessary we finally arrive at the following result.

Theorem 13 Given the exponentiated kernel Fourier matrices $[\widehat{\kappa}(\lambda)]_{\geq[\dots]_{n-k}}$, the kernel (13) can be computed in Fourier space in

$$k(k-1) \sum_{\ell=0}^k \binom{2k}{\ell}^2 \ell! + \sum_{\ell=0}^k \binom{k}{\ell}^2 \ell! = O((2k)^{2k+3})$$

operations, including computing the $\{[\widehat{\Omega}(\lambda)]_{\geq[\dots]_{n-k}}\}_{\lambda \in \Lambda_{n-k}^n}$ matrices appearing in (19).

It should be stressed that $(2k)^{2k+3}$ is only an upper bound on the cost of $\{[\widehat{\Omega}(\lambda)]_{\geq[\dots]_{n-k}}\}_{\lambda \in \Lambda_{n-k}^n}$, and in practice, the size of the $[\rho_\lambda(\sigma)]_{\mathcal{T}_{n-2k}^\lambda, \mathcal{T}_{n-2k}^\lambda}$ blocks does not grow as fast as implied by Theorem 13 (see “ c_2 ” in Table 1 for an exact operations count). We have not addressed the complexity of computing $[\widehat{\kappa}(\lambda)]_{\geq[\dots]_{n-k}}$, because these are constant matrices that can be pre-computed before any kernel evaluations take place. For top- k rankings the issue here is computing the diffusion kernel, which is significantly accelerated by Proposition 2. For interleaving rankings the $\widehat{\Pi}_k^n(\lambda)$ also have to be computed. In our experience for n around 10 all this takes just a few minutes. For much larger n , however, some additional computational tricks or approximations will have to be employed.

6 Experiments

Fourier transforms on \mathbb{S}_n can be computed with the open source FFT library `Snob` (Kondor, 2006). However, `Snob` can only manipulate dense Fourier matrices, so to take advantage of the results described in Section 4 it had to be substantially extended.

We tested our kernel K_{adj} on the “sushi” dataset, available at <http://www.kamishima.net>, which is a dataset of 5000 total rankings of $n = 10$ sushi dishes ranked by different individuals. We subsampled the data to produce 500 “3+2” multirankings, i.e., multirankings of the form $(x_{i_1} \succ x_{i_2} \succ x_{i_3}, x_{i_4} \succ x_{i_5})$ and used 80% of the dataset for training and tested on the remaining 20% to see how well we can predict whether a given individual will choose $x_{i_4} \succ x_{i_5}$ or $x_{i_5} \succ x_{i_4}$ given that she ranked the sushis $x_{i_1}, x_{i_2}, x_{i_3}$ in the order $x_{i_1} \succ x_{i_2} \succ x_{i_3}$. To solve this conditional prediction problem we used an SVM, as in the introduction, which in this particular case, because our prediction is just binary ($x_{i_4} \succ x_{i_5}$ vs. $x_{i_5} \succ x_{i_4}$), reduces to ordinary two-class classification.

As a baseline we used the same SVM with a kernel based on the correlation between partial rankings as described in (Kamishima & Akaho, 2006). Our experiments showed that the SVM trained with our method is relatively insensitive to the value of β and the regularization parameter C , attaining about a 20% error rate (with 10-fold cross-validation) in a wide range of parameter space. In contrast, the optimal performance of the baseline kernel was 42% error. While admittedly preliminary, these experiments show that the adjacent transpositions based diffusion kernel is promising for some types of applications.

Using the Fourier method and caching kernel values that have already been computed, generating the entire 1000×1000 Gram matrix on a desktop machine took less than five minutes. The bulk of this time was taken up by exponentiating the kernel and computing the “riffled kernels” $\widehat{\kappa}(\lambda)$, both of which are pre-computations. Our kernel would have no trouble scaling to larger datasets or much larger n , provided that $\widehat{\kappa}(\lambda)$ can be computed ahead of time or approximated in some way.

7 Conclusions

In this paper we argued that kernels methods are a powerful framework for solving a wide variety of learning tasks involving ranking and ordering. To establish such algorithms, we started by defining some canonical kernels on permutations, namely the diffusion kernels induced from transpositions and adjacent transpositions.

The difficulty with such kernels is that in most ranking problems individual training/test examples are not, in fact, total rankings of all n items under consideration, but partial rankings involving only k items, and thus, naively, each kernel evaluation involves summing over many possibilities. The main technical contribution of this paper was to address this computational issue by showing that in Fourier space the kernel can be efficiently computed.

In particular, we showed that the indicator functions of partial rankings are bandlimited, and that this reduces the complexity of kernel evaluations to $O(n^{3k})$. While this result is novel, it is in many ways a natural extension of spectral analysis on permutations developed in e.g. (Diaconis, 1988), and by itself would not be difficult to derive.

Our more surprising result is that by using techniques involving the group algebra, the complexity can be further reduced to $O((2k)^{2k+3})$, which does not involve n at all. This result applies to averaging any right-invariant kernel over partial rankings, not just diffusion kernels. The reason we

elected to use diffusion kernels was (a) because we believe they are a canonical class of kernels on permutations, and (b) because by Proposition 2 their Fourier transform is easy to pre-compute.

In practice the scaling behavior is much better than what is suggested by our $(2k)^{2k+3}$ upper bound. Still, computationally our kernels are limited to partial rankings of order up to about $k = 7$. For problems where k is large, such as in merging long lists of results from different search engines, it would be better to employ a method that reduces partial rankings to binary rankings or employs a scoring function.

The strength of our method is that it is based purely on the algebra of permutations and does not employ any reduction heuristic, which might cover up some of the structure in the data. Thus, it is best suited to relatively small problems where a careful analysis of the data is required, such as the evaluation of social surveys or voting schemes.

Our paper concentrated on just the kernel, rather than any specific algorithm that it is to be plugged into. There is much room for research on the algorithms side, and on quantifying the complexity of the function classes induced by our kernels. More generally, we feel that our results on the spectral structure of partial rankings are relevant to not just the kernels approach, but to other ranking methods, as well.

Acknowledgments

R. K. would like to thank Tony Jebara for his contributions to some of the initial work on kernels on permutations. Much of the present work was done while M. B. was visiting R. K. at the Gatsby Computational Neuroscience Unit, University College London. National ICT Australia is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian National Research Council through the ICT Center of Excellence program.

References

- Ailon, N., Charikar, M., & Newman, A. (2005). Aggregating inconsistent information: Ranking and clustering. *Proceedings of STOC 2005*.
- Ailon, N., & Mohri, M. (2008). An efficient reduction of ranking to classification. *COLT 2008*.
- Balcan, M.-F., Bansal, N., Beygelzimer, A., Coppersmith, D., Langford, J., & Sorkin, G. B. (2008). Robust reductions from ranking to classification. *Mach. Learn.*, *72*, 139–153.
- Clausen, M. (1989). Fast generalized Fourier transforms. *Theor. Comput. Sci.*, 55–63.
- Diaconis, P. (1988). *Group representation in probability and statistics*, vol. 11 of *IMS Lecture Series*. Institute of Mathematical Statistics.
- Fukumizu, K., Sriperumbudur, B. K., Gretton, A., & Schölkopf, B. (2009). Characteristic kernels on groups and semigroups. In *NIPS 2008*, 473–480.
- Helmbold, D. P., & Warmuth, M. K. (2009). Learning permutations with exponential weights. *Journal of Machine Learning Research*, *10*, 1687–1718.
- Huang, J., & Guestrin, C. (2009). Riffled independence for ranked data. In *NIPS 2009*, 799–807.
- Huang, J., Guestrin, C., & Guibas, L. (2009). Fourier theoretic probabilistic inference over permutations. *Journal of Machine Learning Research*, *10*, 997–1070.
- Kamishima, T., & Akaho, S. (2006). Nantonac collaborative filtering: Recommendation based on multiple order responses. *Proceedings of the international workshop on data mining and statistical science*.
- Kondor, R. (2006). $\mathbb{S}_n\text{ob}$: a C++ library for fast Fourier transforms on the symmetric group. Currently available at <http://www.its.caltech.edu/~risi/Snob/>.
- Kondor, R. (2008). *Group theoretical methods in machine learning*. Ph.D. thesis, Columbia University.
- Kondor, R., Howard, A., & Jebara, T. (2007). Multi-object tracking with representations of the symmetric group. *AISTATS 2007*.
- Kondor, R., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. *ICML 2002*.
- Kondor, R., Shervashidze, N., & Borgwardt, K. (2009). The graphlet spectrum. *ICML 2009*.
- Lebanon, G., & Lafferty, J. (2002). Cranking: Combining rankings using conditional probability models on permutations. *ICML 2002*.
- Maslen, D. K. (1998). The efficient computation of Fourier transforms on the symmetric group. *Mathematics of Computation*, *67*, 1121–1147.

Causal Markov condition for submodular information measures

Bastian Steudel
Max Planck Institute for
Mathematics in the Sciences
Leipzig, Germany
steudel@mis.mpg.de

Dominik Janzing
Max Planck Institute for
Biological Cybernetics
Tübingen, Germany
janzing@tuebingen.mpg.de

Bernhard Schölkopf
Max Planck Institute for
Biological Cybernetics
Tübingen, Germany
schoelkopf@tuebingen.mpg.de

Abstract

The causal Markov condition (CMC) is a postulate that links observations to causality. It describes the conditional independences among the observations that are entailed by a causal hypothesis in terms of a directed acyclic graph. In the conventional setting, the observations are random variables and the independence is a statistical one, i.e., the information content of observations is measured in terms of Shannon entropy. We formulate a generalized CMC for any kind of observations on which independence is defined via an arbitrary submodular information measure. Recently, this has been discussed for observations in terms of binary strings where information is understood in the sense of Kolmogorov complexity. Our approach enables us to find computable alternatives to Kolmogorov complexity, e.g., the length of a text after applying existing data compression schemes. We show that our CMC is justified if one restricts the attention to a class of causal mechanisms that is adapted to the respective information measure. Our justification is similar to deriving the statistical CMC from functional models of causality, where every variable is a deterministic function of its observed causes and an unobserved noise term.

Our experiments on real data demonstrate the performance of compression based causal inference.

1 Introduction

Explaining observations in the sense of inferring the underlying causal structure is among the most important challenges of scientific reasoning. In practical applications it is generally accepted that causal conclusions can be drawn from observing the influence of interventions. The more challenging task, however, is to infer causal relations on the basis of non-interventional observations and research in this direction still is considered with skepticism. It is therefore important to thoroughly formalize the assumptions and discuss the conditions under which they are satisfied. For causal reasoning from statistical data, Spirtes et al. (2001) and Pearl (2000) formalized the assumptions under which the task is solvable. With respect to a causal hypothesis in terms of a directed acyclic graph (DAG) the most basic assumption is the causal Markov condition stating that every variable is conditionally independent of its non-descendants, given its parents,

$$x_j \perp\!\!\!\perp nd_j \mid pa_j,$$

for short. Pearl argues that this follows from a “functional model” of causality (or non-linear structure equations), where every node is a deterministic function of its parents pa_j and an unobserved noise term n_j (see Fig. 1), i.e.,

$$x_j = f_j(pa_j, n_j). \quad (1)$$

The causal Markov condition is then a consequence of the statistical independence of the noise terms, which is called *causal sufficiency*. It can be justified by the assumption that every dependence between them requires a common cause (as postulated by Reichenbach (1956)), which should then explicitly appear in the causal model. From a more abstract point of view, condition (1) can be interpreted as saying that the node x_j does not add any more information that is not already contained in the parents and the noise together. If we restrict the assumption to discrete variables, the corresponding information measure can be, for instance, the Shannon entropy, but also other measures could make sense.

In (Janzing & Schölkopf, 2007) the probabilistic setting is generalized to the case where every observation is formalized by a binary string x_j (without any statistical population). The information content of an observation is then measured using Kolmogorov complexity (also “algorithmic information”) which gives

rise to an algorithmic version of (conditional) mutual information. The corresponding functional model is given by a Turing machine that computes the string x_j from its parent strings pa_j and a noise n_j . The algorithmic information theory based approach generalizes the statistical framework since the average algorithmic information content per instance of a sequence of i.i.d. observations converges to the Shannon entropy, but on the other hand observations need not be generated by i.i.d. sampling.

Unfortunately, Kolmogorov complexity is uncomputable and practical causal inference schemes must deal with other measures of information. In Section 2 we define general information measures and show that they induce independence relations that satisfy the semi-graphoid axioms (Section 3). Then, in Section 4, we phrase the causal Markov condition within our general setting and explore under which conditions it is a reasonable postulate. To this end, we formulate an information theoretic version of functional models observing that their decisive feature is that the joint information of a node, its parents and its noise is the same as the joint information of its parents and noise alone. We demonstrate with examples how these functional models restrict the set of allowed causal mechanisms to a certain class (Section 5). We emphasize that the choice of the information measure determines this class and is therefore the essential prior decision (which certainly requires domain knowledge). Thus, when applying our theory to real data, one first has to think about the causal mechanisms to be explored and then design an information measure that is sufficiently “powerful” to detect the generated dependences.

Section 6 discusses a modification for known independence based causal inference that is necessary for those information measures for which conditioning can only decrease dependences. Section 7 describes one of the most important intended applications of our theory, namely information measures based on compression schemes (e.g. Lempel-Ziv). Applications of these measures using the PC algorithm for causal inference to segments of English text demonstrate the strength of causal reasoning that goes beyond already known applications of compression for the purpose of (hierarchical) clustering.

2 General information measures

In this section we define information from an axiomatic point of view and prove properties that will be useful in the derivation of the causal Markov condition. We start by rephrasing the usual concept of measuring statistical dependences. Let \mathcal{X} be a set of discrete-valued random variables and $\Omega := 2^{\mathcal{X}}$ be the set of subsets. For each $A \in \Omega$ let $H(A)$ denote the joint Shannon entropy of the variables in A . For three disjoint sets A, B, C the conditional mutual information between A and B given C then reads

$$I(A : B|C) := H(A \cup C) + H(B \cup C) - H(A \cup B \cup C) - H(C). \quad (2)$$

The set of subsets constitutes a lattice (Ω, \vee, \wedge) with respect to the operations of union and intersection and H can be seen as a function on this lattice¹. We observe that the non-negativity of (2) can be guaranteed if

$$H(D) + H(E) \geq H(D \vee E) + H(D \wedge E),$$

for two sets $D, E \in \Omega$. This *submodularity* condition is known to be true for Shannon entropy (Cover & Thomas, 2006). Motivated by these remarks, we now introduce an abstract information measure defined on the elements of a general lattice. Throughout this paper let (Ω, \wedge, \vee) be a finite lattice and denote by 0 the meet of all of its elements.

Definition 1 (information measure)

We say $R : \Omega \rightarrow \mathbb{R}$ is an information measure if it satisfies the following axioms:

- (1) *normalization*: $R(0) = 0$,
- (2) *monotonicity*: $s \leq t$ implies $R(s) \leq R(t)$ for all $s, t \in \Omega$,
- (3) *submodularity*: $R(s) + R(t) \geq R(s \vee t) + R(s \wedge t)$ for all $s, t \in \Omega$.

Note that submodular functions have been considered in different contexts, see for example (Lovász, 1983; Matus, 1994; Madiman & Tetali, 2008).

Based on R we define a conditional version for all $s, t \in \Omega$ by

$$R(s|t) := R(s \vee t) - R(t).$$

In analogy to (2), R gives rise to the following measure of independence:

Definition 2 (conditional mutual information) For $s, t, u \in \Omega$ the conditional mutual information of s and t given u is defined by

$$I(s : t|u) := R(s \vee u) + R(t \vee u) - R(s \vee t \vee u) - R(u).$$

We say s and t are independent given u or equivalently $s \perp\!\!\!\perp t|u$ if $I(s : t|u) = 0$.

¹Also the information measures that are presented in this paper can all be rephrased as functions on the lattice of subsets it is nevertheless notationally convenient to formulate the theory with respect to general lattices.

Since the join on lattices is associative and commutative, for ease of notation we write $R(s, t, u, \dots)$ instead of $R(s \vee t \vee u \vee \dots)$ as well as $R(S) := R(s_1 \vee \dots \vee s_n)$ for a subset $S = \{s_1, \dots, s_n\} \subseteq \Omega$. Further $I(s_1, \dots, s_n : u)$ is to be read $I((s_1 \vee \dots \vee s_n) : u)$. The following Lemmas generalize usual information theory.

Lemma 1 (non-negativity of mutual information and conditioning) For $s, t, u \in \Omega$ we have

$$(a) \quad I(s : t|u) \geq 0 \quad \text{and} \quad (b) \quad 0 \leq R(s|t, u) \leq R(s|t).$$

Proof: (a) By definition, $I(s : t|u) \geq 0$ is equivalent to $R(s, u) + R(t, u) \geq R(s, t, u) + R(u)$. Defining $a = s \vee u$ and $b = t \vee u$ and using associativity of \vee we have $a \vee b = s \vee t \vee u$. Further, using Lemma 4 in Ch.1 from (Birkhoff, 1995), in any lattice

$$a \wedge b = (s \vee u) \wedge (t \vee u) \geq u \vee (s \wedge t) \geq u$$

and hence by monotonicity of R : $R(a \wedge b) \geq R(u)$. Combining everything

$$R(s, u) + R(t, u) = R(a) + R(b) \geq R(a \vee b) + R(a \wedge b) \geq R(s, t, u) + R(u),$$

where the first inequality uses submodularity of R .

(b) The first inequality follows from (a) by $I(s : s|t, u) \geq 0$. The second inequality follows directly from (a) and the definition of I . \square

Lemma 2 (chain rule for mutual information) For $s, t, u, x \in \Omega$

$$I(s : t \vee u | x) = I(s : t | x) + I(s : u | t, x). \quad (3)$$

Proof: This is directly seen by using the definition of conditional mutual information on both sides. \square

Lemma 3 (data processing inequality) Given $s, t, x \in \Omega$ it holds

$$R(s|t) = 0 \quad \Rightarrow \quad I(s : x | t) = 0 \quad \Rightarrow \quad I(s : x) \leq I(t : x).$$

Proof: The first implication is clear. For the second we apply the chain rule for mutual information two times and obtain

$$I(s : x) = I(s, t : x) - I(t : x | s) = I(t : x) + I(s : x | t) - I(t : x | s) \leq I(t : x),$$

since the second summand is zero by assumption and conditional mutual information is non-negative. \square

3 Submodular dependence measures and semi-graphoid axioms

The axiomatic approach to stochastic independence goes back to Dawid (1979) who stated four axioms of conditional independence that are fulfilled for any kind of probability distribution. Later, any relation I on triplets that satisfies the same axioms has been named semi-graphoid by Pearl (2000). It is easy to see that the function I constructed from R in the last section satisfies these axioms.

Theorem 1 (I satisfies semi-graphoid axioms) The function I defined in the last section satisfies the semi-graphoid axioms, namely for $x, y, w, z \in \Omega$

$$\begin{array}{llll} (1) & I(x : y | z) = 0 & \Rightarrow & I(y : x | z) = 0 \quad (\text{symmetry}) \\ (2) & I(x : y, w | z) = 0 & \Rightarrow & \begin{cases} I(x : y | z) = 0 \\ I(x : w | z) = 0 \end{cases} \quad (\text{decomposition}) \\ (3) & I(x : y, w | z) = 0 & \Rightarrow & I(x : y | z, w) = 0 \quad (\text{weak union}) \\ (4) & \left. \begin{array}{l} I(x : w | z, y) = 0 \\ I(x : y | z) = 0 \end{array} \right\} & \Rightarrow & I(x : w, y | z) = 0 \quad (\text{contraction}) \end{array}$$

Proof: Symmetry is clear and the remaining implications follow directly from the chain rule and non-negativity. \square

On the contrary, if we are given a function $I : \Omega \times \Omega \times \Omega \rightarrow \mathbb{R}_+$, what axioms do we need to define a submodular information measure R from I ? It turns out that the chain rule in eq. (3) together with non-negativity $I(a : b|c) \geq 0$ and symmetry $I(a : b|c) = I(b : a|c)$ already implies that $R(a) := I(a : a|0)$ is an information measure and I coincides with the dependence measure introduced in Definition 2. We omit the proof due to space constraints.

Thus we characterized the type of dependence measures that we are able to incorporate into our framework. To conclude, note that the chain rule is actually a strong restriction. As an example consider the lattice of linear subspaces of some finite vector space, where the join of two subspaces is the subspace generated by the set-theoretic union and the intersection is just the set-theoretic intersection. An independence measure can be defined by

$$I(a : b | c) = \dim(a|_{c^\perp})|_{(b|_{c^\perp})},$$

where $a|_b$ stands for the orthogonal projection of a onto b and c^\perp denotes the orthogonal complement of c . This is a quantitative version of a notion of independence that satisfies the semi-graphoid axioms (Lauritzen, 1996) even though the chain rule does not hold.

4 Causal Markov condition for general information measures

In this section we define three versions of the causal Markov condition with respect to a general submodular information measure and show that they are equivalent (similar to the statistical framework). Then we discuss under which conditions we expect it to be a reasonable postulate that links observations with causality. Assume we are given observations x_1, \dots, x_k that are connected by a DAG. It is no restriction to consider the observations as elements of a lattice, e.g. the lattice of their subsets.

Definition 3 (causal Markov condition (CMC), local version) *Let G be a DAG that describes the causal relations among observations x_1, \dots, x_k . Then the observations are said to fulfill the causal Markov condition with respect to the dependence measure I if*

$$I(nd_j : x_j | pa_j) = 0 \quad \text{for all } 1 \leq j \leq k,$$

where pa_j denotes the join of the parents of x_j and nd_j the join of its non-descendants (excluding the parents).

The intuitive meaning of the postulate is that conditioning on the direct causes of an observation screens off its dependences from all its non-effects. The following theorem generalizes results in (Lauritzen, 1996) for statistical independences and (Janzing & Schölkopf, 2007) for algorithmic independences. In particular it states that if the causal Markov condition holds with respect to a graph G , then independence relations implied by the CMC can be obtained through the convenient graph-theoretical criterion of d-separation (Pearl, 2000; Spirtes et al., 2001). Two sets of nodes A and B of a DAG are d-separated given a set C disjoint from A and B if every undirected path between A and B is blocked by C . A path that is described by the ordered tuple of nodes (x_1, x_2, \dots, x_r) with $x_1 \in A$ and $x_r \in B$ is blocked if at least one of the following is true

- (1) there is an i such that $x_i \in C$ and $x_{i-1} \rightarrow x_i \rightarrow x_{i+1}$ or $x_{i-1} \leftarrow x_i \leftarrow x_{i+1}$ or $x_{i-1} \leftarrow x_i \rightarrow x_{i+1}$,
- (2) there is an i such that x_i and its descendants are not in C and $x_{i-1} \rightarrow x_i \leftarrow x_{i+1}$.

Theorem 2 (Equivalence of Markov conditions and information decomposition) *Let the nodes x_1, \dots, x_k of a DAG G be elements of some lattice Ω and R be an information measure on Ω . Then the following three properties are equivalent*

- (1) x_1, \dots, x_k fulfill the (local) causal Markov condition.
- (2) For every ancestral set² $A \subseteq \{x_1, \dots, x_k\}$, R decomposes according to G :

$$R(A) = \sum_{x_i \in A} R(x_i | pa_i).$$

- (3) The global Markov condition holds, i.e., if two sets of nodes A and B are d-separated in G given a set C disjoint from A and B , then

$$\left(\bigvee_{a \in A} a \right) \perp\!\!\!\perp \left(\bigvee_{b \in B} b \right) \mid \left(\bigvee_{c \in C} c \right).$$

We omit the proof due to space constraints. The second condition shows that the joint information of observations can be recursively computed according to the causal structure. The third condition describes explicitly which sets of independences are implications of the causal Markov condition.

Our next Theorem will show that the CMC follows from a general notion of a functional model. At its basis is the following Lemma describing that the CMC on a given set of observations can be derived from the causal Markov condition with respect to an extended causal graph (see Figure 1).

²A set A of nodes of a DAG G is called ancestral, if for every $v \in A$ the parents of v are in A too.

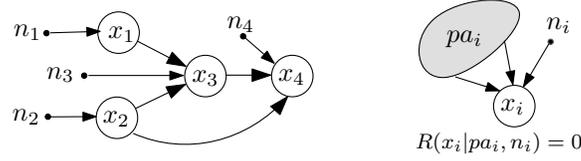


Figure 1: On the left a causal model of four observations x_1, \dots, x_4 is shown together with the 'noise' for each node. In Lemma 4 it is shown that the causal Markov condition on this extended graph implies the CMC for x_1, \dots, x_4 . On the right hand side the functional model assumption is illustrated: The generation of x_i from its parents pa_i and the 'noise' does not produce additional information.

Lemma 4 (causal Markov condition from extended graph) *Let the nodes x_1, \dots, x_k of a DAG G be elements of a lattice Ω with an independence relation I that is monotone and satisfies the chain rule. If there exist additional elements $n_1, \dots, n_k \in \Omega$ such that for all j*

$$I(x_j : nd_j, n_{-j} | pa_j, n_j) = 0, \quad \text{where } n_{-j} = \bigvee_{i \neq j} n_i, \quad (4)$$

and the n_j are jointly independent in the sense that

$$I(n_j : n_{-j}) = 0, \quad (5)$$

then the x_1, \dots, x_k fulfill the causal Markov condition with respect to G .

Proof: Based on G we construct a new graph G' with node set $\{n_1, \dots, n_k\} \cup \{x_1, \dots, x_k\}$ and an additional edge $n_j \rightarrow x_j$ for every j , ($1 \leq j \leq k$). We first show that the causal Markov condition holds for the nodes of G' : By construction, the join of non-descendants nd'_j of x_j with respect to G' is equal to $n_{-j} \vee nd_j$. Since the join of the parents pa'_j of x_j in G' are $pa_j \vee n_j$, assumption (4) just states $I(x_j : nd'_j | pa'_j) = 0$ which is the local CMC with respect to x_j . To see that CMC also holds for n_j , observe that the non-descendants of n_j are equal to the non-descendants of x_j in G' and since n_j does not have any parents, we have to show

$$I(n_j : nd'_j) = 0. \quad (6)$$

Using $nd'_j = n_{-j} \vee nd_j$ together with the chain rule for mutual information we get

$$I(n_j : nd_j, n_{-j}) = I(n_j : n_{-j}) + I(n_j : nd_j | n_{-j}) = I(n_j : nd_j | n_{-j}),$$

where the last equality follows from (5). Let $ND_j = \{x_{j_1}, \dots, x_{j_{k_j}}\}$ be the set of non-descendants of x_j in G . Note that ND_j is ancestral, that is if $x \in ND_j$, then so are the ancestors of x . We introduce a topological order on ND_j , such that if there is an edge $x_{j_a} \rightarrow x_{j_b}$ in G , then $x_{j_a} < x_{j_b}$. Using the chain rule for mutual information iteratively we get

$$I(n_j : nd_j | n_{-j}) = \sum_{a=1}^{k_j} I(n_j : x_{j_a} | x_{j_a}^{(<a)}, n_{-j}),$$

where $x_{j_a}^{(<a)}$ denotes the join of elements of ND_j smaller than x_{j_a} . By choice of our ordering the mutual information of n_j and x_{j_a} is conditioned at least on its parents and we can write $x_{j_a}^{(<a)} = pa_{j_a}^{(<a)} \vee pa_{j_a}^c$, where $pa_{j_a}^c$ is the join of elements smaller than x_{j_a} in ND_j that are not its parents. Therefore, again by the chain rule, each summand on the right hand side can be bounded from above by writing

$$\begin{aligned} I(n_j : x_{j_a} | x_{j_a}^{(<a)}, n_{-j}) &\leq I(n_{-j_a}, pa_{j_a}^c : x_{j_a} | pa_{j_a}, n_{j_a}) \\ &\leq I(n_{-j_a}, nd_{j_a} : x_{j_a} | pa_{j_a}, n_{j_a}) = 0, \end{aligned}$$

where the second inequality is true because by construction $pa_{j_a}^c$ is the join of non-descendants of x_{j_a} . The right hand side vanishes because of assumption (4). This proves (6) and therefore the causal Markov condition with respect to G' .

By Theorem 2, d-separation on G' implies independence. Due to the special structure of G' one can check that d-separation in G implies d-separation in the extended graph G' . Again by Theorem 2, d-separation implies the causal Markov condition for G , which proves the lemma. \square

Now we formalize the intuition that in a generalized functional model a node only contains information that is already contained in the direct causes and the noise together (see Figure 1):

Definition 4 (functional model) Let G be a DAG with nodes x_1, \dots, x_k in the lattice Ω . If there exists an additional node $n_j \in \Omega$ for each x_j , such that the n_j are jointly independent and

$$R(x_j, pa_j, n_j) = R(pa_j, n_j) \quad \text{for all } j, (1 \leq j \leq k) \quad (7)$$

then G together with n_1, \dots, n_k is called a functional model of the x_1, \dots, x_k .

If we restrict our attention to causal mechanism of the above form, the CMC is justified:

Theorem 3 (functional model implies CMC) If there exists a functional model for the nodes x_1, \dots, x_k of a DAG G then they fulfill the causal Markov condition with respect to G .

Proof: In the functional model with noise nodes n_i it holds $R(x_j, pa_j, n_j) = R(pa_j, n_j)$ for all j . This implies $I(nd_j : x_j | pa_j, n_j) = 0$. Since the n_j in a functional model are assumed to be jointly independent, Lemma 4 can be applied and proves the theorem. \square

The following section describes examples of causal mechanisms that can be seen as functional models with respect to various information measures.

5 Examples of information measures and their functional models

Let $S = \{x_1, \dots, x_k\}$ be a finite set of observations which are in a canonical way elements of the lattice of subsets $(2^S, \cup, \cap)$. Let the causal structure be a DAG with x_1, \dots, x_k as nodes.

5.1 Shannon entropy of random variables

Let the x_i be discrete random variables with joint probability mass function $p(x_1, \dots, x_k)$. For a subset $A \subseteq \{x_1, \dots, x_k\}$ denote by $x_A := \times_{x_i \in A} x_i$ the random variable with distribution $p_A := p((x_i)_{x_i \in A})$. The Shannon entropy for the subset A is defined as $H(A) := -\mathbb{E}_p \log p_A$. Monotonicity as well as sub-modularity are well-known properties (Cover & Thomas, 2006). The corresponding notion of independence is the familiar (conditional) stochastic independence, its information-theoretic quantification I being mutual information. Then $H(x_i, pa_i, n_i) = H(pa_i, n_i)$ is equivalent to the existence of some function f_i with

$$x_i = f_i(pa_i, n_i).$$

This restricts the set of mechanisms to those which were deterministic if one could take all latent factors into account. Note that continuous Shannon entropy is not monotone under restriction to subsets. Nevertheless, in this case the chain rule and non-negativity is true and therefore the CMC can be motivated by independences with respect to an extended causal model (Lemma 4 of the previous section).

5.2 Kolmogorov complexity of binary strings

Let the x_i be binary strings and the information measure be the Kolmogorov complexity as information measure. More explicitly, for a subset of strings $A \subseteq S$ denote by x_A a concatenation of the strings in a prefix free manner (which guarantees that the concatenation can be uniquely decoded into its components). The Kolmogorov complexity $K(x_A)$ is then defined as the length of the shortest program that generates the concatenated string x_A on a universal prefix-free Turing machine. It is submodular up to a logarithmic constant (Hammer et al., 2000). For two strings s, t the conditional Kolmogorov complexity $K(s|t)$ of s , given t is defined as the length of the shortest program that computes s from the input t . It must be distinguished from $K(s|t^*)$, the length of the shortest program that computes s from the shortest compression of t . Note that defining $R(s) := K(s)$ implies that the conditional information reads $R(s|t) = K(s|t^*)$ due to (Chaitin, 1975)

$$K(s, t) \stackrel{\pm}{=} K(t) + K(s|t^*),$$

see also (Gács et al., 2001). Then

$$K(x_i, pa_i, n_i) \stackrel{\pm}{=} K(pa_i, n_i) \quad \text{is equivalent to} \quad K(x_i|(pa_i, n_i)^*) \stackrel{\pm}{=} 0,$$

which, in turn, is equivalent to the existence of a program of length $O(1)$ that computes x_i from the shortest compression of (pa_i, n_i) . Here we have considered the number k of nodes as a constant, which ensures that the order of the strings does not matter. Such an ‘‘algorithmic model of causality’’ (Janzing & Schölkopf, 2007) restricts causal influences to *computable* ones. Uncomputable mechanisms can easily be defined (as in the halting problem). However, in the spirit of the Church-Turing thesis, we will assume that they don’t exist in nature and conjecture that the algorithmic model of causality is the most general model of a causal mechanism as long as we restrict the attention to the non-quantum world (where the model could be replaced with a quantum Turing machine).

5.3 Period length of time series

We now present an example of an information measure on a lattice of observations different from the lattice of subsets. Let every observation be a natural number $x_i \in \mathbb{N}$ and consider them elements of the lattice of natural numbers where \vee denotes the least common multiple and \wedge the greatest common divisor, hence for $S \subseteq \{x_1, \dots, x_k\}$

$$x_S := \vee_{x_i \in S} x_i := \text{lcm}(S) \quad .$$

We define an information measure by

$$R(x_S) := \log x_S .$$

Non-negativity and monotonicity of R are clear and submodularity even holds with equality: For $a, b \in \mathbb{N}$

$$\begin{aligned} R(a \vee b) + R(a \wedge b) &= \log \text{lcm}(a, b) + \log \text{gcd}(a, b) = \log \frac{ab}{\text{gcd}(a, b)} + \log \text{gcd}(a, b) \\ &= R(a) + R(b). \end{aligned}$$

The corresponding conditional dependence measure reads

$$I(a : b|c) = R(\text{gcd}(a, b)/\text{gcd}(a, b, c)) = \log \text{gcd}(a, b) - \log \text{gcd}(a, b, c),$$

so a and b are independent given c if c contains all prime factors that are shared by a and b (with at least the same multiplicity).

We define a functional model where every node x_i contains only prime factors that are already contained in its parents and its noise node (with at least the same multiplicity) and the noise terms are assumed to be relatively prime.

Such a lattice of observations can occur in real-life if x_i denotes the period length of a periodic time series over \mathbb{Z} . Then the period length of the joint time series defined by a set of nodes is obviously the least common multiple. If every time series at node i is a function F_i of its parents and noise node (each being a time series) and F_i is *time-covariant*, x_i divides their period lengths.

Assuming that the period lengths of the noise time series are relatively prime is indeed a strong restriction, but if we assume that the periods are large numbers and interpret independence in the approximate sense

$$\log \text{lcm}(x_1, \dots, x_k) \approx \sum_{i=1}^k \log x_i ,$$

we obtain the condition that their periods have no *large* factors in common. This seems to be a reasonable assumption if the noise time series have no common cause.

One can easily think of generalizations where every observation x_i is characterized by a symmetry group and the join of nodes by the group intersection describing the joint symmetry. One may then define functional models where every node inherits all those symmetries that are shared by all its parents and the noise node.

5.4 Size of vocabulary in a text

Let every observation x_i be a text and for every collection of texts $S \subseteq \{x_1, \dots, x_k\}$ let $R(S)$ be the number of different meaningful words in S . Here, meaningful means that we ignore words like articles and prepositions. To see that R is submodular we observe that it is just the number of elements of a set.

We can use R to explore which author has copied parts of the texts written by other authors: Let every x_i be written by another author and a causal arrow from x_i to x_j means that the author of x_i was influenced by x_i when writing x_j .

The noise n_i can be interpreted as the set of words the author usually uses and the condition $R(x_i, pa_i, n_i) = R(pa_i, n_i)$ then means that he/she combines only words from the texts he/she has seen with the own vocabulary.

To conclude this section we want to emphasize that the above example refers to a dependence measure that is non-increasing under conditioning, that is for collections S, T, U and V of texts $I(S : T|U) \geq I(S : T|V)$ whenever $U \subseteq V$. This is because $I(S : T|U)$ is equal to the number of meaningful words contained in S and T , but not in U .³ We will elaborate on this point in the next section because it imposes special challenges for causal inference.

³In general, the above information measure can be viewed as rank or height function on the lattice of sets of meaningful words and it can be shown that dependence measures originating from information functions that are rank functions on distributive lattices are always non-increasing under conditioning.

6 Faithfulness for monotone dependence measures

Apart from the CMC, the essential postulate of independence based causal inference is usually *causal faithfulness*. It states that all observed independence relations are structural, that is, they are induced by the true causal DAG through d-separation. This postulate allows the identification of causal DAGs up to “Markov equivalence classes” imposing the same independences.

Faithfulness has already been defined for abstract conditional independence statements and we start by rephrasing the definition following (Spirtes et al. (2001), p.81).

Definition 5 (faithfulness) *A DAG G is said to represent a set of conditional independence relations \mathcal{L} on a set of observations X faithfully, if \mathcal{L} consists exactly of the independence relations implied by G through d-separation. Further, a set of observations X is said to be faithful (w.r.t. a given dependence measure), if there exists a causal DAG that represents X faithfully.*

The above definition of faithfulness makes sense for the probabilistic and algorithmic notions of dependence, but there is a problem with respect to dependence measures on which conditioning can only decrease information. As mentioned above, rank functions of distributive lattices lead to this kind of dependence measures, that we will call *monotone* in the following. To see the problem, consider for three observations a, b, c a causal DAG G of the form $a \rightarrow b \leftarrow c$. By d-separation, a is independent of c and for a monotone dependence measure this implies $a \perp\!\!\!\perp c|b$, which is not an independence induced by d-separation. Hence, G does not faithfully represent the objects and one can easily check that a faithful representation does not exist (e.g. using the theorem below). However, we can modify faithfulness such that it also accounts for those independences that follow from monotonicity under conditioning:

Definition 6 (monotone faithfulness) *A DAG G is said to represent a set \mathcal{L} of conditional independences of observations X monotonically faithful, if the following condition is true for all disjoint subsets $S, T, U \subseteq X$ whose join is denoted by s, t and u : Whenever $s \perp\!\!\!\perp t|u$ is in \mathcal{L} and u is minimal among all the sets that render s and t independent, then s and t are d-separated by u in G . Further, a set of observations X is said to be monotonically faithful (w.r.t. a given dependence measure), if there exists a causal DAG that represents X monotonically faithful.*

Note that, trivially, every faithful representation is a monotonically faithful representation, hence faithful observations are monotonically faithful observations. Faithful representations have already been characterized (Theorem 3.4 in (Spirtes et al., 2001)) and we prove an equivalent characterization that holds simultaneously for monotonically faithful and for faithful observations.

Theorem 4 (characterization of monotonically faithful representations) *A set of (monotonically) faithful observations X is represented (monotonically) faithfully by a DAG G if and only if (1) and (2) holds, where:*

- (1) *two observations a and b are adjacent in G if and only if they can not be made independent by conditioning on any join of observations in $X \setminus \{a, b\}$.*
- (2) *for three observations a, b, c , such that a is adjacent to b , b is adjacent to c and a is not adjacent to c , it holds that $a \rightarrow b \leftarrow c$ in G if and only if there exists a set $U \subseteq X \setminus \{a, b, c\}$ such that a is independent of c given the join of the observations in U .*

We omit the proof due to space constraints. The theorem implies in particular, that every monotonically faithful representation of faithful objects is already a faithful representation.

The PC algorithm (Spirtes & Glymour, 1991; Spirtes et al., 2001) for causal inference takes a set of conditional independences on faithful objects and returns the equivalence class of faithful representations. Since the above theorem is used to prove the correctness of the algorithm in the faithful case, we conclude that the algorithm correctly returns monotonically faithful representations given monotonically faithful observations. We apply the PC-algorithm with respect to compression based information functions in the following section. Also they are not monotone in a strict theoretical sense, empirical observations indicate that it is unlikely for the mutual information to increase.

7 Compression based information

In this section we demonstrate that our framework enables us to do causal inference on single *objects* (coded as binary strings) without relying on the uncomputable measure of Kolmogorov complexity. To this end, instead of defining complexity with respect to a universal Turing machine we explicitly limit ourselves to specific production processes of strings. The underlying measure of information is motivated by universal compression algorithms like LZ77 (Ziv & Lempel, 1977) and grammar based compression (Yang & Kieffer, 2000) that detect repeated occurrences of identical substrings within a given input string and encode them

more efficiently. The choice of a compression scheme can be seen as a prior analogously to the choice of a universal Turing machine in the case of algorithmic information. The measures considered in this section quantify the information of an observation (string) in terms of the diversity of its substrings and entail the following assumption on causal processes: A mechanism that produces a string y from a string x is considered as simple, if it constructs y by concatenating a small number of substrings from x (see Lemma 6 below for a formal statement). Further, the amount of dependence of observations is approximately given by the number of substrings that they share.

We are going to describe two specific measures of information that are closely related to the total length of the compressed string, but have better formal properties than the latter. This way our conclusions will be independent of the actual implementation of the compression scheme and proving theoretical results gets easier.

In the last part of this section we describe experiments on real data in which the PC algorithm is applied to infer the causal structure using either of the two introduced measures of information.

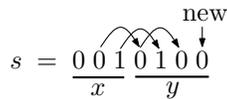
Note that *distance metrics* based on compression length have already been used to cluster various kinds of data (see (Cilibiasi & Vitányi, 2005) for computable distance metrics motivated by algorithmic mutual information or (Hanus et al., 2007) for an application to molecular biology). These metrics can be used to reconstruct trees (hierarchical clustering) but if two nodes are linked by more than one path a measure of conditional mutual information is needed to reconstruct the data-generation process. To the best of our knowledge, methods that rely on compression based *conditional* mutual information have not been used before to infer non-tree-like DAGs.

7.1 Lempel-Ziv information (LZ-information)

LZ-information has been introduced as a complexity measure for strings in (Ziv & Lempel, 1976). It has been applied to quantify the complexity of time series in biomedical signal analysis (Aboy et al., 2006) and distance measures based on versions of LZ-information have been used to analyze neural spike train data (Blanc et al., 2008) and to reconstruct phylogenetic trees (Zhen et al., 2009). We start by defining

Definition 7 (production and reproduction from prefix) *Let $s = xy$ be a string. We say s is reproducible from its prefix x and write $x \rightarrow s$ if y is a substring of $x\bar{y}$, where \bar{y} is equal to y without its last symbol. We say s is producible from x and write $x \Rightarrow s$ if $x \rightarrow \bar{s}$, where \bar{s} is equal to s without its last symbol.*

Contrary to reproducibility, producibility allows for the generation of new substrings, for if $x \Rightarrow s$, the last symbol of s can be arbitrary.



Example: For a given string $s = xy$ let \bar{s} be the string without its last symbol. The figure on the left shows that \bar{s} is producible from its prefix x by copying the second symbol of x to the first of y and so on. The string s itself is not producible from x , but reproducible.

Informally, LZ-information counts the minimal number of times during the process of parsing the input string from left to right, in which the string can not be reproduced from its prefix and a production step is needed.

Definition 8 (LZ-information, (Ziv & Lempel, 1976)) *Let s be a string of length n . Denote by s_i the i -th symbol of s and by $s(i, j)$ the substring $s_i s_{i+1} \dots s_j$. A production history H_s of s is a partition of s into substrings $s = s(h_0, h_1) s(h_1 + 1, h_2) \dots s(h_k + 1, h_{k+1})$ with $h_0 = 1$ and $h_{k+1} = n$, such that*

$$s(1, h_i) \Rightarrow s(1, h_{i+1}) \quad \text{for all } i \in \{1, \dots, k\}.$$

A history H_s is called exhaustive if additionally

$$s(1, h_i) \not\Rightarrow s(1, h_{i+1}) \quad \text{for all } i \in \{1, \dots, k-1\}.$$

The substrings $s(h_i + 1, h_{i+1})$, ($0 \leq i \leq k$) will be called components of H_s and the length $|H_s|$ of H_s is defined as the number of its components.

The LZ-information of s , denoted by $c(s)$, is defined as the length of its (unique) exhaustive history.

In an exhaustive history, each h_i is chosen maximal such that $s(1, h_i - 1)$ is reproducible from its prefix $s(1, h_{i-1})$. As an example, for $s = 000100101100110$ the exhaustive history partitions s into

$$s = (0)(001)(00101)(10011)(0),$$

hence $c(s) = 5$.

In the original paper of Ziv and Lempel (1976) it was shown that c is subadditive: for two strings x and y the information of the concatenated string xy is at most the information of x plus the information of y . This already suggests to define the non-negative unconditional dependency measure $i(x : y) = c(x) + c(y) - c(xy)$. As it turns out, non-negativity of conditional information holds up to a negligible constant independent of the involved string lengths:

Lemma 5 (non-negativity of conditional LZ-information, asymmetric version) *Let x, y, z be finite strings over some alphabet \mathcal{A} . Further let α and β be symbols not contained in \mathcal{A} that will be used as separators. Then*

$$i(x : y|z) := c(z\alpha x) + c(z\alpha y) - c(z\alpha x\beta y) - c(z) \geq -1. \quad (8)$$

Proof: Let $E_{z\alpha}$ be the exhaustive history of $z\alpha$. The exhaustive history of $z\alpha x$ is of the form $E_{z\alpha x} = [E_{z\alpha}, E_{x|z}]$, where $E_{x|z}$ describes the partition of x induced by $E_{z\alpha x}$. This is because α is not part of the alphabet, hence the component in $E_{z\alpha x}$ containing α must be of the form $(t\alpha)$ for some substring t . Analogously $E_{z\alpha y} = [E_{z\alpha}, E_{y|z}]$. It is not difficult to see that

$$H_{z\alpha x\beta y} = [E_{z\alpha}, E_{x|z}, \beta, E_{y|z}].$$

is a production history of $z\alpha x\beta y$. Theorem 1 in (Ziv & Lempel, 1976) states that a production history is at least as long as the exhaustive history, hence

$$|[E_{z\alpha}, E_{x|z}, \beta, E_{y|z}]| \geq |E_{z\alpha x\beta y}| = c(z\alpha x\beta y),$$

Further, $c(z) \leq |E_{z\alpha}|$ and so (8) can be bounded from below by

$$\begin{aligned} c(z\alpha x) + c(z\alpha y) - c(z\alpha x\beta y) - c(z) &\geq |[E_{z\alpha}, E_{x|z}]| + |[E_{z\alpha}, E_{y|z}]| - |[E_{z\alpha}, E_{x|z}, \beta, E_{y|z}]| - |E_{z\alpha}| \\ &= -1. \end{aligned}$$

□

The above Lemma shows, that for two sets of strings $A = \{z, x\}$ and $B = \{z, y\}$ the LZ-information of $A \cup B$ and $A \cap B$ (represented by the information of strings $z\alpha x\beta y$ and z) exceeds the LZ-information of A and B (represented by the information of the strings $z\alpha x$ and $z\alpha y$) by at most one. This can be interpreted as approximate 'submodularity' with respect to A and B .

Within the functional models introduced before a node x_i was assumed to contain at most as much information as its parents pa_i and an independent noise n_i . The following Lemma states that if x_i is produced by concatenating complex substrings of pa_i and n_i , this is approximately the case with respect to LZ-information.

Lemma 6 (functional model for LZ-information, asymmetric version) *Let pa_i and n_i be two strings over an alphabet \mathcal{A} and construct a third string x_i by concatenating k substrings of pa_i and n_i . Then*

$$c(pa_i \alpha n_i \beta x_i) \leq c(pa_i \alpha n_i \beta) + k,$$

where α and β are symbols not in \mathcal{A} used as separators.

Proof: A production history of $pa_i \alpha n_i \beta x_i$ can be generated by concatenating the exhaustive history of $pa_i \alpha n_i \beta$ with the list of the at most k substrings out of which x_i is constructed. The length of this history is $c(pa_i \alpha n_i \beta) + k + 1$ and bounds $c(pa_i \alpha n_i \beta x_i)$ from above by Theorem 1 in (Ziv & Lempel, 1976). □

In particular, if xy is producible from x , by appending y , the information is at most increased by one. Hence, if we restrict the mechanisms that generate a node to consist of a limited number of concatenations of substrings from its parents and the independent noise (compared to the amounts of information involved) the causal Markov condition would follow if c were an information function. This is not the case since c is not defined on sets of strings (in particular it is not symmetric ($c(xy) \neq c(yx)$)), therefore we define the LZ-information of a set of strings to be the LZ-information of their concatenation with respect to a given order (e.g. lexicographic).

Definition 9 (LZ-information, set version) *Let $\{x_1, \dots, x_k\}$ be a set of strings over some alphabet \mathcal{A} . Choose k distinct symbols $\alpha_1, \dots, \alpha_k$ not contained in \mathcal{A} that will be used as separators.*

Let $X = \{x_{i_1}, \dots, x_{i_m}\}$ be a subset and assume $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_m}$ with respect to a given order on the set of strings over \mathcal{A} . We define the LZ-information of X as

$$LZ(X) = c(x_{i_1} \alpha_{i_1} \dots x_{i_m} \alpha_{i_m}),$$

where the argument of c is understood as the concatenation of the strings.

LZ is not monotone and submodular in a strict sense. However, empirical observations suggest that for sufficiently large strings the violations of submodularity induced by the asymmetries like $c(x\alpha y) \neq c(y\alpha x)$ are negligible compared to the amounts of information.

Hypothesis: For practical purposes $LZ(\cdot)$ is an information measure up to constants that are negligible compared to the amounts of information of the strings involved. The associated independence measure I is monotonically decreasing (through conditioning).

We close by mentioning that the calculation of the LZ-information is very inefficient for large strings since one has to search over all substrings of the part of the string already parsed. In our implementation we therefore considered only substrings of length limited by a constant (we chose 30 for strings of English text, since it is unlikely that a substring of length 30 is repeated exactly).

7.2 Grammar based information

In the grammar based approach to compression an input string x is transformed into a context-free grammar that generates x . This grammar is then compressed for example using arithmetic codes. We discuss this approach because it has been successfully applied to compress RNA data (e.g. (Liu et al., 2008)). Further the LZ-based compression discussed in the previous section can be rephrased into this framework. As there are many grammars that produce a given string, it is essential that the transformation of strings to grammars produces economic representations of x (for an overview see (Lehman & Shelat, 2002)) We implemented the so called greedy grammar transform from Yang and Kieffer (2000). It constructs the grammar iteratively by parsing the input string x . Due to space restrictions we just give an example of a string and its generated grammar.

Example: The binary string $x = 1001110001000$ is transformed using the greedy grammar transform from Yang and Kieffer (2000) to the grammar $G(x)$:

$$\begin{aligned} s_0 &\rightarrow s_1 11 s_2 s_2 \\ s_1 &\rightarrow 100 \\ s_2 &\rightarrow s_1 0, \end{aligned}$$

where s_0, s_1 and s_2 are variables of the grammar and x can be reconstructed by starting from s_0 and then iteratively substituting s_i by the right hand side of each production rule above. The *length of a grammar* $|G(x)|$ is defined as the sum of all symbols on the right of every production rule, so for the above example $|G(x)| = 10$. We view the length of the constructed grammar as information measure of the string that it produces and define analog to the LZ-information

Definition 10 (grammar based information) Let $\{x_1, \dots, x_k\}$ be a set of strings over some alphabet \mathcal{A} . Choose k distinct symbols $\alpha_1, \dots, \alpha_k$ not contained in \mathcal{A} that will be used as separators.

Let $X = \{x_{i_1}, \dots, x_{i_m}\}$ be a subset and assume $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_m}$ with respect to a given order on the set of strings over \mathcal{A} . We define the grammar based information of X as

$$GR(X) = |G(x_{i_1} \alpha_{i_1} \dots x_{i_m} \alpha_{i_m})|,$$

where the input of the grammar construction G is understood as the concatenation of the strings.

By definition GR is non-negative. However, experiments show that submodularity is violated, but the amount of violation still allows to draw causal conclusions for sufficiently large strings.

7.3 Experiments

This section reports the results on causal inference using the introduced LZ-information and grammar based information measures. Matlab code of the algorithms used in the experiments can be downloaded from the homepage of the first author.

Experiment 1: Markov chains of English texts

We start with a string of English text s_0 from which we construct further strings s_1, \dots, s_k as follows: To generate s_{i+1} we translate s_i using an automatic translator from Google⁴ to a randomly chosen European language. Then s_{i+1} is defined as the string that we obtain when we translate s_i back to English using the same translator. Since s_{i+1} is *determined* by s_i , the process can be modeled by a 'Markov' chain $s_0 \rightarrow \dots \rightarrow s_k$. We then apply the PC algorithm⁵ to infer the corresponding equivalence class of (monotonically) faithful causal models consisting of the DAGs:

$$s_0 \leftarrow \dots \leftarrow s_i \rightarrow \dots \rightarrow s_k \quad \text{for } 0 \leq i \leq k.$$

In our experiments we chose several starting texts of 1000 to 5000 symbols (e.g. news articles and the abstract of this paper) and generated three strings ($k = 3$) using the described procedure. In every string we transformed all non-space characters to numbers $0, \dots, 8$ using a modulo operation on the ASCII value to reduce the alphabet size. Repeated spaces were deleted and the space character has been encoded separately by the number 9 to ensure that words of the string remain separated.

Results: Based on the two information measures, the PC algorithm returned the correct class of DAGs in every case. For LZ-information the chosen threshold used to determine independence did not even have to depend on the starting texts s_0 . Grammar based information seems to be more sensitive to the string lengths involved and we had to choose a different threshold for every chosen text s_0 . Further, we successfully tried the method on the chain of preliminary versions of the abstract of this paper.

⁴accessible at <http://translate.google.de/>

⁵Our implementation of the PC algorithm for causal inference was based on the BNT-Toolbox for Matlab written by Kevin Murphy and available at <http://code.google.com/p/bnt/>.

Finally note that methods based on compression distance could also be applied to recover the correct equivalence class. The crucial difference to our approach consists in the fact that we did not have to assume that the underlying graph is a tree.

Experiment 2: Four-node networks

We want to infer the equivalence classes of (monotonically) faithful causal models depicted in Figures (a) and (b) below. To this end we randomly choose segments of a large English text and then construct the strings corresponding to the nodes a, b, c and d in a way that ensures the resulting observation $\{a, b, c, d\}$ to be (monotonically) faithful. Explicitely, we choose segments s_x and s_{xy} for each node x and for each edge between nodes x and y respectively. Further, for every ordered triple of nodes (x, y, z) whose subgraph is not equal to $x \rightarrow y \leftarrow z$, we pick a segment s_{xyz} . This way we obtain the following segments with respect to the graph in Figure (a):

$$s_a, s_b, s_c, s_d, s_{ab}, s_{ac}, s_{bd}, s_{cd}, s_{bac}, s_{abd}, s_{acd}$$

and with respect to the graph in Figure (b) we get segments

$$s_a, s_b, s_c, s_d, s_{ac}, s_{ad}, s_{bc}, s_{bd}, s_{cad}, s_{cbd}.$$

Finally, the string at a node is constructed as the concatenation of all segments that contain the name of the node in its index (the order is arbitrary), e.g. in the case of Figure (a)

$$b = s_b s_{ab} s_{bd} s_{bac} s_{abd}.$$

As text source we chose an English version of Anna Karenina by Lev Tolstoj⁶. We then transformed all non-space characters to numbers from $0, \dots, 8$ using a modulo operation on the ASCII value to reduce the size of the alphabet. Repeated spaces were deleted and the space character has been encoded separately by the number 9 to ensure that words of the string remain separated. The resulting string consisted of a total of approximately two million symbols. Using the above construction, we generated 100 observations $\{a, b, c, d\}$ with respect to each graph and applied the PC algorithm. The length N of the randomly chosen segments was chosen uniformly between 100 and 200 in the first run and between 300 and 500 in the second run. The choice of the threshold to determine independence depended only on the information measure and on the two possible ranges of N , but not on the individual observations. Further, the graph of Figure (b) implies an unconditional independence of a and b . Since two disjoint segments of English text can not be expected to be independent, we conditioned all informations that we calculate on background knowledge in terms of fixed segment of length 5000.

(a)		Correct answers of PC:		(b)		Correct answers of PC:	
		$N \in [100, 200]$				$N \in [100, 200]$	
		LZ :	98%			LZ :	95%
		GR :	53%			GR :	97%
	$N \in [300, 500]$			$N \in [300, 500]$			
	LZ :	100%		LZ :	100%		
	GR :	56%		GR :	99%		

Results: Above, the percentages of correct results from the PC-algorithm are shown. Note that using LZ-information we were able to recover the correct equivalence class in almost all runs independently of the graph structure and segment length. Grammar based inference did not perform quite as well, but in the majority of cases in which it did not return the correct Markov equivalence class most of the independences still were detected correctly.

8 Conclusions

We have introduced conditional dependence measures that originate from submodular measures of information. We argued that these notions of conditional dependence (generalizing statistical dependence) can be used to infer the causal structure among observations even if the latter are not generated by i.i.d. sampling. To this end, we formulated a generalized causal Markov condition (with significant formal analogies to the statistical one) and proved that the condition is justified provided that the attention is restricted to a class of causal mechanisms that depends on the underlying measure of information. We demonstrated that existing compression schemes like Lempel-Ziv define interesting notions of information and described the class of mechanisms that justify the causal Markov condition in this case. Accordingly, we showed that the PC-algorithm successfully infers causal relations among texts when based on a notion of dependence that is induced by compression schemes.

⁶The text is available at <http://www.gutenberg.org/etext/1399>.

References

- Aboy, M., Hornero, R., Abasolo, D., & Alvarez, D. (2006). Interpretation of the Lempel-Ziv Complexity Measure in the Context of Biomedical Signal Analysis. *IEEE Transactions on Biomedical Engineering*, 53, issue 11, 2282–2288.
- Birkhoff, G. (1995). *Lattice theory*. American Mathematical Society. 3rd edition.
- Blanc, J.-L., Schmidt, N., Bonnier, L., Pezard, L., & Lesne, A. (2008). Quantifying neural correlations using Lempel-Ziv complexity. *Deuxieme conference francaise de Neurosciences Computationnelles*.
- Chaitin, G. J. (1975). A theory of program size formally identical to information theory. *J. ACM*, 22, 329–340.
- Cilibrasi, R., & Vitányi, P. M. B. (2005). Clustering by compression. *IEEE Transactions on Information Theory*, 51, 1523–1545.
- Cover, T. M., & Thomas, J. A. (2006). *Elements of information theory*. Wiley-Interscience. 2nd edition.
- Dawid, A. P. (1979). Conditional independence in statistical theory. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41, 1–31.
- Gács, P., Tromp, J. T., & Vitányi, P. M. (2001). Algorithmic statistics. *IEEE Transactions on Information Theory*, 47, 2443–2463.
- Hammer, D., Romashchenko, A., Shen, A., & Vereshchagin, N. (2000). Inequalities for Shannon entropy and Kolmogorov complexity. *Journal of Computer and System Sciences*, 60, 442 – 464.
- Hanus, P., Dingel, J., Zech, J., Hagenauer, J., & Müller, J. C. (2007). Information theoretic distance measures in phylogenomics. *Proc. International Workshop on Information Theory and Applications (ITA 2007)*.
- Janzing, D., & Schölkopf, B. (2007). Causal inference using the algorithmic Markov condition. <http://arxiv.org/abs/0804.3678>, to appear in *IEEE Transactions on Information Theory*.
- Lauritzen, S. L. (1996). *Graphical models*. Oxford Statistical Science Series. Oxford University Press, USA.
- Lehman, E., & Shelat, A. (2002). Approximation algorithms for grammar-based compression. In *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms* (pp. 205–212). ACM/SIAM.
- Liu, Q., Yang, Y., Chen, C., Bu, J., Zhang, Y., & Ye, X. (2008). RNACompress: Grammar-based compression and informational complexity measurement of RNA secondary structure. *BMC Bioinformatics*, 9, 176.
- Lovász, L. (1983). Submodular functions and convexity. *Mathematical Programming—The State of the Art*, 22, 235–257.
- Madiman, M., & Tetali, P. (2008). Information inequalities for joint distributions, with interpretations and applications.
- Matus, F. (1994). Probabilistic conditional independence structures and matroid theory: Background. *Int. J. of General Systems*, 22, 185–196.
- Pearl, J. (2000). *Causality*. Cambridge University Press.
- Reichenbach, H. (1956). *The direction of time*. University of California Press.
- Spirites, P., & Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9, 62–72.
- Spirites, P., Glymour, C., & Scheines, R. (2001). *Causation, prediction, and search, second edition (adaptive computation and machine learning)*. The MIT Press.
- Yang, E.-H., & Kieffer, J. C. (2000). Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform. *IEEE Transactions on Information Theory*, 46, 755–777.
- Zhen, X., Li, C., & Wang, J. (2009). A complexity-based measure and its application to phylogenetic analysis. *Journal of Mathematical Chemistry*, 4, 1149–1157.
- Ziv, J., & Lempel, A. (1976). On the complexity of finite sequences. *IEEE Transactions on Information Theory*, 22, 75–81.
- Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23, 337–343.

Open Loop Optimistic Planning

Sébastien Bubeck, Rémi Munos
SequeL Project, INRIA Lille
40 avenue Halley,
59650 Villeneuve d'Ascq, France
{sebastien.bubeck, remi.munos}@inria.fr

Abstract

We consider the problem of planning in a stochastic and discounted environment with a limited numerical budget. More precisely, we investigate strategies exploring the set of possible sequences of actions, so that, once all available numerical resources (e.g. CPU time, number of calls to a generative model) have been used, one returns a recommendation on the best possible immediate action to follow based on this exploration. The performance of a strategy is assessed in terms of its simple regret, that is the loss in performance resulting from choosing the recommended action instead of an optimal one. We first provide a minimax lower bound for this problem, and show that a uniform planning strategy matches this minimax rate (up to a logarithmic factor). Then we propose a UCB (Upper Confidence Bounds)-based planning algorithm, called OLOP (Open-Loop Optimistic Planning), which is also minimax optimal, and prove that it enjoys much faster rates when there is a small proportion of near-optimal sequences of actions. Finally, we compare our results with the regret bounds one can derive for our setting with bandits algorithms designed for an infinite number of arms.

1 Introduction

We consider the problem of planning in general stochastic and discounted environments. More precisely, the decision making problem consists in an exploration phase followed by a recommendation. First, the agent explores freely the set of possible sequences of actions (taken from a finite set A of cardinality K), using a finite budget of n actions. Then the agent makes a recommendation on the first action $a(n) \in A$ to play. This decision making problem is described precisely in Figure 1. The goal of the agent is to find the best way to explore its environment (first phase) so that, once the available resources have been used, he is able to make the best possible recommendation on the action to play in the environment.

During the exploration of the environment, the agent iteratively selects sequences of actions, under the global constraint that he can not take more than n actions in total, and receives a reward after each action. More precisely, at time step t during the m^{th} sequence, the agent played $a_{1:t}^m = a_1^m \dots a_t^m \in A^t = A \times \dots \times A$ and receives a discounted reward $\gamma^t Y_t^m$ where $\gamma \in (0, 1)$ is the discount factor. We make a stochastic assumption on the generating process for the reward: given $a_{1:t}^m$, Y_t is drawn from a probability distribution $\nu(a_{1:t}^m)$ on $[0, 1]$. Given $a \in A^t$, we write $\mu(a)$ for the mean of the probability $\nu(a)$.

The performance of the recommended action $a(n) \in A$ is assessed in terms of the so-called **simple regret** r_n , which is the performance loss resulting from choosing this sequence and then following an optimal path instead of following an optimal path from the beginning:

$$r_n = V - V(a(n)),$$

where $V(a(n))$ is the (discounted) value of the action (or sequence) $a(n)$, defined for any finite sequence of actions $a \in A^h$ as:

$$V(a) = \sup_{u \in A^\infty : u_{1:h} = a} \sum_{t \geq 1} \gamma^t \mu(u_{1:t}), \quad (1)$$

and V is the optimal value, that is the maximum expected sum of discounted rewards one may obtain (i.e. the sup in (1) is taken over all sequences in A^∞).

Note that this simple regret criterion has already been studied in multi-armed bandit problems, see Bubeck et al. (2009a); Audibert et al. (2010).

Exploration in a stochastic and discounted environment.

Parameters available to the agent: discount factor $\gamma \in (0, 1)$, number of actions K , number of rounds n .

Parameters unknown to the agent: the reward distributions $\nu(a)$, $a \in A^*$.

For each episode $m \geq 1$; for each moment in the episode $t \geq 1$;

- (1) If n actions have already been performed then the agent outputs an action $a(n) \in A$ and the game stops.
- (2) The agent chooses an action $a_t^m \in A$.
- (3) The environment draws $Y_t^m \sim \nu(a_{1:t}^m)$ and the agent receives the reward $\gamma^t Y_t^m$.
- (4) The agent decides to either move the next moment $t + 1$ in the episode or to reset to its initial position and move the next episode $m + 1$.

Goal: maximize the value of the recommended action (or sequence): $V(a(n))$ (see (1) for the definition of the value of an action).

Figure 1: Exploration in a stochastic and discounted environment.

An important application of this framework concerns the problem of planning in Markov Decision Processes (MDPs) with very large state spaces. We assume that the agent possesses a generative model which enables to generate a reward and a transition from any state-action to a next state, according to the underlying reward and transition model of the MDP. In this context, we propose to use the generative model to perform a planning from the current state (using a finite budget of n calls to a generative model) to generate a near-optimal action $a(n)$ and then apply $a(n)$ in the real environment. This action modifies the environment and the planning procedure is repeated from the next state to select the next action and so on. From each state, the planning consists in the exploration of the set of possible sequences of actions as described in Figure 1, where the generative model is used to generate the rewards.

Note that, using control terminology, the setting described above (from a given state) is called “open-loop” planning, because the class of considered policies (i.e. sequences of actions) are only function of time (and not of the underlying resulting states). This open-loop planning is in general sub-optimal compared to the optimal (closed-loop) policy (mapping from states to actions). However, here, while the planning is open-loop (i.e. we do not take into consideration the subsequent states in the planning), the resulting general policy is closed-loop (since the chosen action depends on the current state).

This approach to MDPs has already been investigated as an alternative to usual dynamic programming approaches (which approximate the optimal value function to design a near optimal policy) to circumvent the computational complexity issues. For example, Kearns et al. describe a sparse sampling method that uses a finite amount of computational resources to build a look-ahead tree from the current state, and returns a near-optimal action with high probability.

Another field of application is POMDPs (Partially Observable Markov Decision Problems), where from the current belief state an open-loop plan may be built to select a near-optimal immediate action (see e.g. Yu et al. (2005); Hsu et al. (2007)). Note that, in these problems, it is very common to have a limited budget of computational resources (CPU time, memory, number of calls to the generative model, ...) to select the action to perform in the real environment, and we aim at making an efficient use of the available resources to perform the open-loop planning.

Moreover, in many situations, the generation of state-transitions is computationally expensive, thus it is critical to make the best possible use of the available number of calls to the model to output the action. For instance, an important problem in waste-water treatment concerns the control of a biochemical process for anaerobic digestion. The chemical reactions involve hundreds of different bacteria and the simplest models of the dynamics already involve dozens of variables (for example, the well-known model called ADM1 Batstone et al. (2002) contains 32 state variables) and their simulation is numerically heavy. Because of the curse of dimensionality, it is impossible to compute an optimal policy for such model. The methodology described above aims at a less ambitious goal, and search for a closed-loop policy which is open-loop optimal at each time step. While this policy is suboptimal, it is also a more reasonable target in terms of computational complexity. The strategy considered here proposes to use the model to simulate transitions and perform a complete open-loop planning at each time step.

The main contribution of the paper is the analysis of an adaptive exploration strategy of the search space, called Open-Loop Optimistic Planning (OLOP), which is based on the “optimism in the face of uncertainty” principle, i.e. where the most promising sequences of actions are explored first. The idea of optimistic planning has already been investigated in the simple case of deterministic environments, Hren and Munos (2008). Here we consider the non-trivial extension of this optimistic approach to planning in stochastic environments. For that purpose, upper confidence bounds (UCBs) are assigned to all sequences of actions, and the exploration expands further the sequences with highest UCB. The idea of selecting actions based on UCBs comes from the multi-armed bandits literature, see Lai and Robbins (1985); Auer et al. (2002). Planning under uncertainty using UCBs has been considered previously in Chang et al. (2007) (the so-called UCB sampling) and in Kocsis and Szepesvari (2006), where the resulting algorithm, UCT (UCB applied to Trees), has been successfully applied to the large scale tree search problem of computer-go, see Gelly et al. (2006). However, its regret analysis shows that UCT may perform very poorly because of overly-optimistic assumptions in the design of the bounds, see Coquelin and Munos (2007). Our work is close in spirit to BAST (Bandit Algorithm for Smooth Trees), Coquelin and Munos (2007), the Zooming Algorithm, Kleinberg et al. (2008) and HOO (Hierarchical Optimistic Optimization), Bubeck et al. (2009b). Like in these previous works, the performance bounds of OLOP are expressed in terms of a measure of the proportion of near-optimal paths.

However, as we shall discuss in Section 4, these previous algorithms fail to obtain minimax guarantees for our problem. Indeed, a particularity of our planning problem is that the value of a sequence of action is defined as the sum of discounted rewards along the path, thus the rewards obtained along any sequence provides information, not only about that specific sequence, but also about any other sequence sharing the same initial actions. OLOP is designed to use this property as efficiently as possible, to derive tight upper-bounds on the value of each sequence of actions.

Note that our results does not compare with traditional regret bounds for MDPs, such as the ones proposed in Auer et al. (2009). Indeed, in this case one compares to the optimal closed-loop policy, and the resulting regret usually depends on the size of the state space (as well as on other parameters of the MDP).

Outline. We exhibit in Section 2 the minimax rate (up to a logarithmic factor) for the simple regret in discounted and stochastic environments: both lower and upper bounds are provided. Then in Section 3 we describe the OLOP strategy, and show that if there is a small proportion of near-optimal sequences of actions, then faster rates than minimax can be derived. In Section 4 we compare our results with previous works and present several open questions. Finally the Appendix contains the analysis of OLOP.

Notations To shorten the equations we use several standard notations over alphabets. We collect them here: $A^0 = \{\emptyset\}$, A^* is the set of finite words over A (including the null word \emptyset), for $a \in A^*$ we note $h(a)$ the integer such that $a \in A^{h(a)}$, $aA^h = \{ab, b \in A^h\}$, for $a \in A^h$ and $h' > h$ we note $a_{1:h'} = a\emptyset \dots \emptyset$ and $a_{1:0} = \emptyset$.

2 Minimax optimality

In this section we derive a lower bound on the simple regret (in the worst case) of any agent, and propose a simple (uniform) forecaster which attains this optimal minimax rate (up to a logarithmic factor). The main purpose of the section on the uniform planning is to show explicitly the special concentrations property that our model enjoys.

2.1 Minimax lower bound

We propose here a new lower bound, whose proof follows from a simple adaptation of the technique developed in Auer et al. (2003). Note that this lower bound is not a particular case of the ones derived in Kleinberg et al. (2008) or Bubeck et al. (2009b) in a more general framework, as we shall see in Section 4.

Theorem 1 *Any agent satisfies:*

$$\sup_{\nu} \mathbb{E}r_n = \begin{cases} \Omega\left(\left(\frac{\log n}{n}\right)^{\frac{\log 1/\gamma}{\log K}}\right) & \text{if } \gamma\sqrt{K} > 1, \\ \Omega\left(\sqrt{\frac{\log n}{n}}\right) & \text{if } \gamma\sqrt{K} \leq 1. \end{cases}$$

2.2 Uniform Planning

To start gently, let us consider first (and informally) a *naive version* of the uniform planning. One can choose a depth H , uniformly test all sequences of actions in A^H (with $(n/H)/K^H$ samples for each sequence), and

then return the empirical best sequence. Cutting the sequences at depth H implies an error of order γ^H , and relying on empirical estimates with $(n/H)/K^H$ samples adds an error of order $\sqrt{\frac{HK^H}{n}}$, leading to a simple regret bounded as $O\left(\gamma^H + \sqrt{\frac{HK^H}{n}}\right)$. Optimizing over H yields an upper bound on the simple regret of the naive uniform planning of order:

$$O\left(\left(\frac{\log n}{n}\right)^{\frac{\log 1/\gamma}{\log K + 2 \log 1/\gamma}}\right), \quad (2)$$

which does not match the lower bound. The cautious reader probably understands why this version of uniform planning is suboptimal. Indeed we do not use the fact that any sequence of actions of the form ab gives information on the sequences ac . Hence, the concentration of the empirical mean for short sequences of actions is much faster than for long sequences. This is the critical property which enables us to fasten the rates with respect to traditional methods, see Section 4 for more discussion on this.

We describe now the *good version* of uniform planning. Let $H \in \mathbb{N}$ be the largest integer such that $HK^H \leq n$. Then the procedure goes as follows: For each sequence of actions $a \in A^H$, the uniform planning allocates one episode (of length H) to estimate the value of the sequence a , that is it receives $Y_t^a \sim \nu(a_{1:t})$, $1 \leq t \leq H$ (drawn independently). At the end of the allocation procedure, it computes for all $a \in A^h$, $h \leq H$, the empirical average reward of the sequence a :

$$\hat{\mu}(a) = \frac{1}{K^{H-h}} \sum_{b \in A^H: b_{1:h}=a} Y_h^b.$$

(obtained with K^{H-h} samples.) Then, for all $a \in A^H$, it computes the empirical value of the sequence a :

$$\hat{V}(a) = \sum_{t=1}^H \gamma^t \hat{\mu}(a_{1:t}).$$

It outputs $a(n) \in A$ defined as the first action of the sequence $\arg \max_{a \in A^H} \hat{V}(a)$ (ties break arbitrarily).

This version of uniform planning makes a much better use of the reward samples than the naive version. Indeed, for any sequence $a \in A^h$, it collects the rewards Y_h^b received for sequences $b \in aA^{H-h}$ to estimate $\mu(a)$. Since $|aA^{H-h}| = K^{H-h}$, we obtain an estimation error for $\mu(a)$ of order $\sqrt{K^{h-H}}$. Then, thanks to the discounting, the estimation error for $V(a)$, with $a \in A^H$, is of order $K^{-H/2} \sum_{h=1}^H (\gamma\sqrt{K})^h$. On the other hand, the approximation error for cutting the sequences at depth H is still of order γ^H . Thus, since H is the largest depth (given n and K) at which we can explore once each node, we obtain the following behavior: When K is large, precisely $\gamma\sqrt{K} > 1$, then H is small and the estimation error is of order γ^H , resulting in a simple regret of order $n^{-(\log 1/\gamma)/\log K}$. On the other hand, if γ is small, precisely $\gamma\sqrt{K} < 1$, then the depth H becomes less important, and the estimation error is of order $K^{-H/2}$, resulting in a simple regret of order $n^{-1/2}$. This reasoning can easily be made precise to prove the following Theorem.

Theorem 2 *The (good) uniform planning satisfies:*

$$\mathbb{E}r_n \leq \begin{cases} O\left(\sqrt{\log n} \left(\frac{\log n}{n}\right)^{\frac{\log 1/\gamma}{\log K}}\right) & \text{if } \gamma\sqrt{K} > 1, \\ O\left(\frac{(\log n)^2}{\sqrt{n}}\right) & \text{if } \gamma\sqrt{K} = 1, \\ O\left(\frac{\log n}{\sqrt{n}}\right) & \text{if } \gamma\sqrt{K} < 1. \end{cases}$$

Remark 1 *We do not know whether the $\sqrt{\log n}$ (respectively $(\log n)^{3/2}$ in the case $\gamma\sqrt{K} = 1$) gap between the upper and lower bound comes from a suboptimal analysis (either in the upper or lower bound) or from a suboptimal behavior of the uniform forecaster.*

3 OLOP (Open Loop Optimistic Planning)

The uniform planning described in Section 2.2 is a static strategy, it does not adapt to the rewards received in order to improve its exploration. A stronger strategy could select, at each round, the next sequence to explore as a function of the previously observed rewards. In particular, since the value of a sequence is the sum

of discounted rewards, one would like to explore more intensively the sequences starting with actions that already yielded high rewards. In this section we describe an adaptive exploration strategy, called Open Loop Optimistic Planning (OLOP), which explores first the most promising sequences, resulting in much stronger guarantees than the one derived for uniform planning.

OLOP proceeds as follows. It assigns upper confidence bounds (UCBs), called B-values, to all sequences of actions, and selects at each round a sequence with highest B-value. This idea of a UCB-based exploration comes from the multi-armed bandits literature, see Auer et al. (2002). It has already been extended to hierarchical bandits, Chang et al. (2007); Kocsis and Szepesvari (2006); Coquelin and Munos (2007), and to bandits in metric (or even more general) spaces, Auer et al. (2007); Kleinberg et al. (2008); Bubeck et al. (2009b).

Like in these previous works, we express the performance of OLOP in terms of a measure of the proportion of near-optimal paths. More precisely, we define $\kappa_c \in [1, K]$ as the branching factor of the set of sequences in A^h that are $c \frac{\gamma^{h+1}}{1-\gamma}$ -optimal, where c is a positive constant, i.e.

$$\kappa_c = \limsup_{h \rightarrow \infty} \left| \left\{ a \in A^h : V(a) \geq V - c \frac{\gamma^{h+1}}{1-\gamma} \right\} \right|^{1/h}. \quad (3)$$

Intuitively, the set of sequences $a \in A^h$ that are $\frac{\gamma^{h+1}}{1-\gamma}$ -optimal are the sequences for which the perfect knowledge of the discounted sum of mean rewards $\sum_{t=1}^h \gamma^t \mu(a_{1:t})$ is not sufficient to decide whether a belongs to an optimal path or not, because of the unknown future rewards for $t > h$. In the main result, we consider κ_2 (rather than κ_1) to account for an additional uncertainty due to the empirical estimation of $\sum_{t=1}^h \gamma^t \mu(a_{1:t})$. In Section 4, we discuss the link between κ and the other measures of the set of near-optimal states introduced in the previously mentioned works.

3.1 The OLOP algorithm

The OLOP algorithm is described in Figure 2. It makes use of some B-values assigned to any sequence of actions in A^L . At time $m = 0$, the B-values are initialized to $+\infty$. Then, after episode $m \geq 1$, the B-values are defined as follows: For any $1 \leq h \leq L$, for any $a \in A^h$, let

$$T_a(m) = \sum_{s=1}^m \mathbb{1}\{a_{1:h}^s = a\}$$

be the number of times we played a sequence of actions beginning with a . Now we define the empirical average of the rewards for the sequence a as:

$$\hat{\mu}_a(m) = \frac{1}{T_a(m)} \sum_{s=1}^m Y_h^s \mathbb{1}\{a_{1:h}^s = a\},$$

if $T_a(m) > 0$, and 0 otherwise. The corresponding upper confidence bound on the value of the sequence of actions a is by definition:

$$U_a(m) = \sum_{t=1}^h \left(\gamma^t \hat{\mu}_{a_{1:t}}(m) + \gamma^t \sqrt{\frac{2 \log M}{T_{a_{1:t}}(m)}} \right) + \frac{\gamma^{h+1}}{1-\gamma},$$

if $T_a(m) > 0$ and $+\infty$ otherwise. Now that we have upper confidence bounds on the value of many sequences of actions we can sharpen these bounds for the sequences $a \in A^L$ by defining the B-values as:

$$B_a(m) = \inf_{1 \leq h \leq L} U_{a_{1:h}}(m).$$

At each episode $m = 1, 2, \dots, M$, OLOP selects a sequence $a^m \in A^L$ with highest B-value, observes the rewards $Y_t^m \sim \nu(a_{1:t}^m)$, $t = 1, \dots, L$ provided by the environment, and updates the B-values. At the end of the exploration phase, OLOP returns an action that has been the most played, i.e. $a(n) = \operatorname{argmax}_{a \in A} T_a(M)$.

3.2 Main result

Theorem 3 (Main Result) *Let $\kappa_2 \in [1, K]$ be defined by (3). Then, for any $\kappa' > \kappa_2$, OLOP satisfies:*

$$\mathbb{E}r_n = \begin{cases} \tilde{O}\left(n^{-\frac{\log 1/\gamma}{\log \kappa'}}\right) & \text{if } \gamma\sqrt{\kappa'} > 1, \\ \tilde{O}\left(n^{-\frac{1}{2}}\right) & \text{if } \gamma\sqrt{\kappa'} \leq 1. \end{cases}$$

(We say that $u_n = \tilde{O}(v_n)$ if there exists $\alpha, \beta > 0$ such that $u_n \leq \alpha(\log(v_n))^\beta v_n$)

Open Loop Optimistic Planning:

Let M be the largest integer such that $M \lceil \log M / (2 \log 1/\gamma) \rceil \leq n$. Let $L = \lceil \log M / (2 \log 1/\gamma) \rceil$.

For each episode $m = 1, 2, \dots, M$;

- (1) The agent computes the B -values at time $m - 1$ for sequences of actions in A^L (see Section 3.1) and chooses a sequence that maximizes the corresponding B -value:

$$a^m \in \operatorname{argmax}_{a \in A^L} B_a(m - 1).$$

- (2) The environment draws the sequence of rewards $Y_t^m \sim \nu(a_{1:t}^m)$, $t = 1, \dots, L$.

Return an action that has been the most played: $a(n) = \operatorname{argmax}_{a \in A} T_a(M)$.

Figure 2: Open Loop Optimistic Planning

Remark 2 *One can see that the rate proposed for OLOP greatly improves over the uniform planning whenever there is a small proportion of near-optimal paths (i.e. κ is small). Note that this does not contradict the lower bound proposed in Theorem 1. Indeed κ provides a description of the environment ν , and the bounds are expressed in terms of that measure, one says that the bounds are distribution-dependent. Nonetheless, OLOP does not require the knowledge of κ , thus one can take the supremum over all $\kappa \in [1, K]$, and see that it simply replaces κ by K , proving that OLOP is minimax optimal (up to a logarithmic factor).*

Remark 3 *In the analysis of OLOP, we relate the simple regret to the more traditional **cumulative regret**, defined at round n as $R_n = \sum_{m=1}^M (V - V(a^m))$. Indeed, in the proof of Theorem 3, we first show that $r_n = \tilde{O}(\frac{R_n}{n})$, and then we bound (in expectation) this last term. Thus the same bounds apply to $\mathbb{E}R_n$ with a multiplicative factor of order n . In this paper, we focus on the simple regret rather than on the traditional cumulative regret because we believe that it is a more natural performance criterion for the planning problem considered here. However note that OLOP is also minimax optimal (up to a logarithmic factor) for the cumulative regret, since one can also derive lower bounds for this performance criterion using the proof of Theorem 1.*

Remark 4 *One can also see that the analysis carries over to $r_n^L = V - V(\operatorname{argmax}_{a \in A^L} T_a(M))$, that is we can bound the simple regret of a sequence of actions in A^L rather than only the first action $a(n) \in A$. Thus, using n actions for the exploration of the environment, one can derive a plan of length L (of order $\log n$) with the optimality guarantees of Theorem 3.*

4 Discussion

In this section we compare the performance of OLOP with previous algorithms that can be adapted to our framework. This discussion is summarized in Figure 3. We also point out several open questions raised by these comparisons.

Comparison with Zooming Algorithm/HOO: In Kleinberg et al. (2008) and Bubeck et al. (2009b), the authors consider a very general version of stochastic bandits, where the set of arms \mathcal{X} is a metric space (or even more general spaces in Bubeck et al. (2009b)). When the underlying mean-payoff function is 1-Lipschitz with respect to the metric (again, weaker assumption are derived in Bubeck et al. (2009b)), the authors propose two algorithms, respectively the Zooming Algorithm and HOO, for which they derive performances in terms of either the zooming dimension or the near-optimality dimension. In a metric space, both of these notions coincide, and the corresponding dimension d is defined such that the number of balls of diameter ε required to cover the set of arms that are ε -optimal is of order ε^{-d} . Then, for both algorithms, one obtains a simple regret of order $\tilde{O}(n^{-1/(d+2)})$ (thanks to Remark 3).

Up to minor details, one can see our framework as a A^∞ -armed bandit problem, where the mean-payoff function is the sum of discounted rewards. A natural metric ℓ on this space can be defined as follows: For any $a, b \in A^\infty$, $\ell(a, b) = \frac{\gamma^{h(a,b)+1}}{1-\gamma}$, where $h(a, b)$ is the maximum depth $t \geq 0$ such that $a_{1:t} = b_{1:t}$. One can very easily check that the sum of discounted reward is 1-Lipschitz with respect to that metric, since

$\sum_{t \geq 1} \gamma^t |\mu(a_{1:t}) - \mu(b_{1:t})| = \sum_{t \geq h(a,b)+1} \gamma^t |\mu(a_{1:t}) - \mu(b_{1:t})| \leq \ell(a, b)$. We show now that κ_2 , defined by (3), is closely related to the near-optimality dimension. Indeed, note that the set aA^∞ can be seen as a ball of diameter $\frac{\gamma^{h(a)+1}}{1-\gamma}$. Thus, from the definition of κ_2 , the number of balls of diameter $\frac{\gamma^{h+1}}{1-\gamma}$ required to cover the set of $2\frac{\gamma^{h+1}}{1-\gamma}$ -optimal paths is of order of κ^h , which implies that the near-optimality dimension is $d = \frac{\log \kappa}{\log 1/\gamma}$. Thanks to this result, we can see that applying the Zooming Algorithm or HOO in our setting yield a simple regret bounded as:

$$\mathbb{E}r_n = \tilde{O}(n^{-1/(d+2)}) = \tilde{O}(n^{-\frac{\log 1/\gamma}{\log \kappa + 2 \log 1/\gamma}}). \quad (4)$$

Clearly, this rate is always worse than the ones in Theorem 3. In particular, when one takes the supremum over all κ , we find that (4) gives the same rate as the one of naive uniform planning in (2). This was expected since these algorithms do not use the specific shape of the global reward function (which is the sum of rewards obtained along a sequence) to generalize efficiently across arms. More precisely, they do not consider the fact that a reward sample observed for an arm (or sequence) ab provides strong information about any arm in aA^∞ . Actually, the difference between HOO and OLOP is the same as the one between the naive uniform planning and the good one (see Section 2.2).

However, although things are obvious for the case of uniform planning, in the case of OLOP, it is much more subtle to prove that it is indeed possible to collect enough reward samples along sequences $ab, b \in A^*$ to deduce a sharp estimation of $\mu(a)$. Indeed, for uniform planning, if each sequence $ab, b \in A^h$ is chosen once, then one may estimate $\mu(a)$ using K^h reward samples. However in OLOP, since the exploration is expected to focus on promising sequences rather than being uniform, it is much harder to control the number of times a sequence $a \in A^*$ has been played. This difficulty makes the proof of Theorem 3 quite intricate compared to the proof of HOO for instance.

Comparison with UCB-AIR: When one knows that there are many near-optimal sequences of actions (i.e. when κ is close to K), then one may be convinced that among a certain number of paths chosen uniformly at random, there exists at least one which is very good with high probability. This idea is exploited by the UCB-AIR algorithm of Wang et al. (2009), designed for infinitely many-armed bandits, where at each round one chooses either to sample a new arm (or sequence in our case) uniformly at random, or to re-sample an arm that has already been explored (using a UCB-like algorithm to choose which one). The regret bound of Wang et al. (2009) is expressed in terms of the probability of selecting an ε -optimal sequence when one chooses the actions uniformly at random. More precisely, the characteristic quantity is β such that this probability is of order of ε^β . Again, one can see that κ_2 is closely related to β . Indeed, our assumption says that the proportion of ε -optimal sequences of actions (with $\varepsilon = 2\frac{\gamma^{h+1}}{1-\gamma}$) is $O(\kappa^h)$, resulting in $\kappa = K\gamma^\beta$. Thanks to this result, we can see that applying UCB-AIR in our setting yield a simple regret bounded as:

$$\mathbb{E}r_n = \begin{cases} \tilde{O}(n^{-\frac{1}{2}}) & \text{if } \kappa > K\gamma \\ \tilde{O}(n^{-\frac{1}{1+\beta}}) = \tilde{O}(n^{-\frac{\log 1/\gamma}{\log K/\kappa + \log 1/\gamma}}) & \text{if } \kappa \leq K\gamma \end{cases}$$

As expected, UCB-AIR is very efficient when there is a large proportion of near-optimal paths. Note that UCB-AIR requires the knowledge of β (or equivalently κ).

Figure 3 shows a comparison of the exponents in the simple regret bounds for OLOP, uniform planning, UCB-AIR, and Zooming/HOO (in the case $K\gamma^2 > 1$). We note that the rate for OLOP is better than UCB-AIR when there is a small proportion of near-optimal paths (small κ). Uniform planning is always dominated by OLOP and corresponds to a minimax lower bound for any algorithm. Zooming/HOO are always strictly dominated by OLOP and they do not attain minimax performances.

Comparison with deterministic setting: In Hren and Munos (2008), the authors consider a deterministic version of our framework, precisely they assume that the rewards are a deterministic function of the sequence of actions. Remarkably, in the case $\kappa\gamma^2 > 1$, we obtain the same rate for the simple regret as Hren and Munos (2008). Thus, in this case, we can say that planning in stochastic environments is not harder than planning in deterministic environments (moreover, note that in deterministic environments there is no distinction between open-loop and closed-loop planning).

Open questions: We identify four important open questions. (i) Is it possible to attain the performances of UCB-AIR when κ is unknown? (ii) Is it possible to improve OLOP if κ is known? (iii) Can we combine the advantages of OLOP and UCB-AIR to derive an exploration strategy with improved rate in intermediate cases (i.e. when $1/\gamma^2 < \kappa < \gamma K$)? (iv) What is a problem-dependent lower bound (in terms of κ or other measures of the environment) in this framework? Obviously these problems are closely related, and the current behavior of the bounds suggests that question (iv) might be tricky. As a side question, note that OLOP requires the knowledge of the time-horizon n , we do not know whether it is possible to obtain the same guarantees with an anytime algorithm.

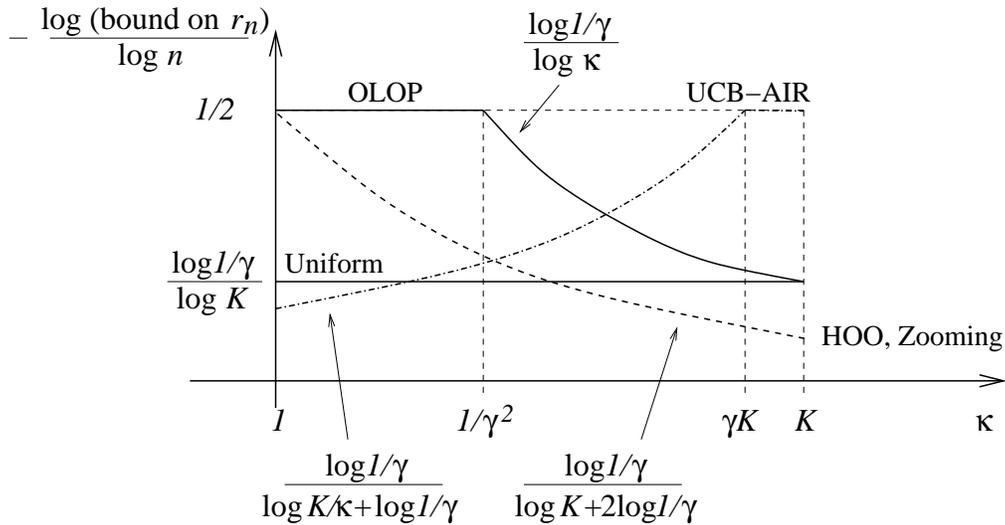


Figure 3: Comparison of the exponent rate of the bounds on the simple regret for OLOP, (good) uniform planning, UCB-AIR, and Zooming/HOO, as a function of $\kappa \in [1, K]$, in the case $K\gamma^2 > 1$.

References

- J.-Y. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. In *23rd annual conference on learning theory*, 2010.
- P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning Journal*, 47(2-3):235–256, 2002.
- P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2003.
- P. Auer, R. Ortner, and C. Szepesvári. Improved rates for the stochastic continuum-armed bandit problem. *20th Conference on Learning Theory*, pages 454–468, 2007.
- P. Auer, T. Jaksch, and R. Ortner. Near-optimal regret bounds for reinforcement learning. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Lon Bottou, editors, *Advances in Neural Information Processing Systems 21*, page 89–96. MIT Press, 2009.
- D. J. Batstone, J. Keller, I. Angelidaki, S. V. Kalyuzhnyi, S. G. Pavlostathis, A. Rozzi, W. T. M. Sanders, H. Siegrist, and V. A. Vavilin. Anaerobic digestion model no. 1 (adm1). *IWA Publishing*, 13, 2002.
- S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *Proc. of the 20th International Conference on Algorithmic Learning Theory*, 2009a.
- S. Bubeck, R. Munos, G. Stoltz, and Cs. Szepesvari. Online optimization in \mathcal{X} -armed bandits. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 201–208, 2009b.
- Hyeong Soo Chang, Michael C. Fu, Jiaqiao Hu, and Steven I. Marcus. *Simulation-based Algorithms for Markov Decision Processes*. Springer, London, 2007.
- P.-A. Coquelin and R. Munos. Bandit algorithms for tree search. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, 2007.
- J. L. Doob. *Stochastic Processes*. John Wiley & Sons, 1953.
- S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with patterns in Monte-Carlo go. Technical Report RR-6062, INRIA, 2006.
- W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

- J.-F. Hren and R. Munos. Optimistic planning for deterministic systems. In *European Workshop on Reinforcement Learning*, 2008.
- D. Hsu, W.S. Lee, and N. Rong. What makes some POMDP problems easy to approximate? In *Neural Information Processing Systems*, 2007.
- M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markovian decision processes. In *Machine Learning*, volume 49, year = 2002,, pages 193–208.
- R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the 40th ACM Symposium on Theory of Computing*, 2008.
- L. Kocsis and Cs. Szepesvari. Bandit based Monte-carlo planning. In *Proceedings of the 15th European Conference on Machine Learning*, pages 282–293, 2006.
- T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.
- Y. Wang, J.Y. Audibert, and R. Munos. Algorithms for infinitely many-armed bandits. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1729–1736. 2009.
- C. Yu, J. Chuang, B. Gerkey, G. Gordon, and A.Y. Ng. Open loop plans in POMDPs. Technical report, Stanford University CS Dept., 2005.

Appendix. Proof of Theorem 3

The proof of Theorem 3 is quite subtle. To present it in a gentle way we adopt a pyramidal proof rather than a pedagogic one. We propose seven lemmas, which we shall not motivate in depth, but prove in details. The precise architecture of the proof is as follows: Lemma 4 is a preliminary step, it justifies Remark 3. Then Lemma 5 underlines the important cases that we have to treat to show that suboptimal arms are not pulled too often. Lemma 6 takes care of one of these cases. Then, from Lemma 7 to 10, each Lemma builds on its predecessor. The main result eventually follows from Lemma 4 and 10 together with a simple optimization step.

We introduce first a few notations that will be useful. Let $1 \leq H \leq L$ and $a^* \in A^L$ such that $V(a^*) = V$. We define now some useful sets for any $1 \leq h \leq H$ and $0 \leq h' < h$;

$$\mathcal{I}_0 = \{\emptyset\}, \quad \mathcal{I}_h = \left\{ a \in A^h : V - V(a) \leq \frac{2\gamma^{h+1}}{1-\gamma} \right\}, \quad \mathcal{J}_h = \{ a \in A^h : a_{1:h-1} \in \mathcal{I}_{h-1} \text{ and } a \notin \mathcal{I}_h \}.$$

Note that, from the definition of κ_2 , we have that for any $\kappa' > \kappa_2$, there exists a constant C such that for any $h \geq 1$,

$$|\mathcal{I}_h| \leq C\kappa'. \quad (5)$$

Now for $1 \leq m \leq M$, and $a \in A^t$ with $t \leq h$, write

$$\mathcal{P}_{h,h'}^a(m) = \left\{ b \in aA^{h-t} \cap \mathcal{J}_h : T_b(m) \geq \frac{8}{\gamma^2} (h+1)^2 \gamma^{2(h'-h)} \log M + 1 \right\}.$$

Finally we also introduce the following random variable:

$$\tau_{h,h'}^a(m) = \mathbb{1} \left\{ T_a(m-1) < \frac{8}{\gamma^2} (h+1)^2 \gamma^{2(h'-h)} \log M + 1 \leq T_a(m) \right\}.$$

Lemma 4 *The following holds true,*

$$r_n \leq \frac{2K\gamma^{H+1}}{1-\gamma} + \frac{3K}{M} \sum_{h=1}^H \sum_{a \in \mathcal{J}_h} \frac{\gamma^h}{1-\gamma} T_a(M).$$

Proof: Since $a(n) \in \arg \max_{a \in A} T_a(M)$, and $\sum_{a \in A} T_a(M) = M$, we have $T_{a(n)}(M) \geq M/K$, and thus:

$$\frac{M}{K} \left(V - V(a(n)) \right) \leq \left(V - V(a(n)) \right) T_{a(n)}(M) \leq \sum_{m=1}^M V - V(a^m).$$

Hence, we have, $r_n \leq \frac{K}{M} \sum_{m=1}^M V - V(a^m)$. Now remark that, for any sequence of actions $a \in A^L$, we have either:

- $a_{1:H} \in \mathcal{I}_H$; which implies $V - V(a) \leq \frac{2\gamma^{H+1}}{1-\gamma}$.
- or there exists $1 \leq h \leq H$ such that $a_{1:h} \in \mathcal{J}_h$; which implies $V - V(a) \leq V - V(a_{1:h-1}) + \frac{\gamma^h}{1-\gamma} \leq \frac{3\gamma^h}{1-\gamma}$.

Thus we can write:

$$\begin{aligned} \sum_{m=1}^M (V - V(a^m)) &= \sum_{m=1}^M (V - V(a^m)) \left(\mathbf{1}\{a^m \in \mathcal{I}_H\} + \mathbf{1}\{\exists 1 \leq h \leq H : a_{1:h}^m \in \mathcal{J}_h\} \right) \\ &\leq \frac{2\gamma^{H+1}}{1-\gamma} M + 3 \sum_{h=1}^H \sum_{a \in \mathcal{J}_h} \frac{\gamma^h}{1-\gamma} T_a(M), \end{aligned}$$

which ends the proof of Lemma 4. ■

The rest of the proof is devoted to the analysis of the term $\mathbb{E} \sum_{a \in \mathcal{J}_h} T_a(M)$. In the stochastic bandit literature, it is usual to bound the expected number of times a suboptimal action is pulled by the inverse suboptimality (of this action) squared, see for instance Auer et al. (2002) or Bubeck et al. (2009b). Specialized to our setting, this implies a bound on $\mathbb{E} T_a(M)$, for $a \in \mathcal{J}_h$, of order γ^{-2h} . However, here, we obtain much stronger guarantees, resulting in the faster rates. Namely we show that $\mathbb{E} \sum_{a \in \mathcal{J}_h} T_a(M)$ is of order $(\kappa')^h$ (rather than $(\kappa')^h \gamma^{-2h}$ with previous methods).

The next lemma describes under which circumstances a suboptimal sequence of actions in \mathcal{J}_h can be selected.

Lemma 5 *Let $0 \leq m \leq M - 1$, $1 \leq h \leq L$ and $a \in \mathcal{J}_h$. If $a^{m+1} \in aA^*$ then it implies that one the three following propositions is true:*

$$\exists 1 \leq h' \leq L : U_{a_{1:h'}}^*(m) < V, \quad (6)$$

or

$$\sum_{t=1}^h \gamma^t \hat{\mu}_{a_{1:t}}(m) \geq V(a) + \sum_{t=1}^h \gamma^t \sqrt{\frac{2 \log M}{T_{a_{1:t}}(m)}}, \quad (7)$$

or

$$2 \sum_{t=1}^h \gamma^t \sqrt{\frac{2 \log M}{T_{a_{1:t}}(m)}} > \frac{\gamma^{h+1}}{1-\gamma}. \quad (8)$$

Proof: If $a^{m+1} \in aA^*$ then it implies that $U_a(m) \geq \inf_{1 \leq h' \leq L} U_{a_{1:h'}}^*(m)$. That is either (6) is true or

$$U_a(m) = \sum_{t=1}^h \gamma^t \hat{\mu}_{a_{1:t}}(m) + \gamma^t \sqrt{\frac{2 \log M}{T_{a_{1:t}}(m)}} + \frac{\gamma^{h+1}}{1-\gamma} \geq V.$$

In the latter case, if (7) is not satisfied, it implies

$$V(a) + 2 \sum_{t=1}^h \gamma^t \sqrt{\frac{2 \log M}{T_{a_{1:t}}(m)}} + \frac{\gamma^{h+1}}{1-\gamma} > V. \quad (9)$$

Since $a \in \mathcal{J}_h$ we have $V - V(a) - \frac{\gamma^{h+1}}{1-\gamma} \geq \frac{\gamma^{h+1}}{1-\gamma}$ which shows that equation (9) implies (8) and ends the proof. ■

We show now that both equations (6) and (7) have a vanishing probability of being satisfied.

Lemma 6 *The following holds true, for any $1 \leq h \leq L$ and $m \leq M$,*

$$\mathbb{P}(\text{equation (6) or (7) is true}) \leq m(L+h)M^{-4} = \tilde{O}(M^{-3}).$$

Proof: Since $V \leq \sum_{t=1}^h \gamma^t \mu(a_{1:t}^*) + \frac{\gamma^{h+1}}{1-\gamma}$, we have,

$$\begin{aligned} & \mathbb{P}(\exists 1 \leq h \leq L : U_{a_{1:h}^*}(m) \leq V) \\ & \leq \mathbb{P}\left(\exists 1 \leq h \leq L : \sum_{t=1}^h \gamma^t \left(\hat{\mu}_{a_{1:t}^*}(m) + \sqrt{\frac{2 \log M}{T_{a_{1:t}^*}(m)}} \right) \leq \sum_{t=1}^h \gamma^t \mu(a_{1:t}^*) \text{ and } T_{a_{1:h}^*}(m) \geq 1\right) \\ & \leq \mathbb{P}\left(\exists 1 \leq t \leq L : \hat{\mu}_{a_{1:t}^*}(m) + \sqrt{\frac{2 \log M}{T_{a_{1:t}^*}(m)}} \leq \mu(a_{1:t}^*) \text{ and } T_{a_{1:t}^*}(m) \geq 1\right) \\ & \leq \sum_{t=1}^L \mathbb{P}\left(\hat{\mu}_{a_{1:t}^*}(m) + \sqrt{\frac{2 \log M}{T_{a_{1:t}^*}(m)}} \leq \mu(a_{1:t}^*) \text{ and } T_{a_{1:t}^*}(m) \geq 1\right). \end{aligned}$$

Now we want to apply a concentration inequality to bound this last term. To do it properly we exhibit a martingale and apply the Hoeffding-Azuma inequality for martingale differences (see Hoeffding (1963)). Let

$$S_j = \min\{s : T_{a_{1:s}^*}(s) = j\}, \quad j \geq 1.$$

If $S_j \leq M$, we define $\tilde{Y}_j = Y_t^{S_j}$, and otherwise \tilde{Y}_j is an independent random variable with law $\nu(a_{1:t}^*)$. We clearly have,

$$\begin{aligned} & \mathbb{P}\left(\hat{\mu}_{a_{1:t}^*}(m) + \sqrt{\frac{2 \log M}{T_{a_{1:t}^*}(m)}} \leq \mu(a_{1:t}^*) \text{ and } T_{a_{1:t}^*}(m) \geq 1\right) \\ & = \mathbb{P}\left(\frac{1}{T_{a_{1:t}^*}(m)} \sum_{j=1}^{T_{a_{1:t}^*}(m)} \tilde{Y}_j + \sqrt{\frac{2 \log M}{T_{a_{1:t}^*}(m)}} \leq \mu(a_{1:t}^*) \text{ and } T_{a_{1:t}^*}(m) \geq 1\right) \\ & \leq \sum_{u=1}^m \mathbb{P}\left(\frac{1}{u} \sum_{j=1}^u \tilde{Y}_j + \sqrt{\frac{2 \log M}{u}} \leq \mu(a_{1:t}^*)\right). \end{aligned}$$

Now we have to prove that $\tilde{Y}_j - \mu(a_{1:t}^*)$ is martingale differences sequence. This follows via an optional skipping argument, see (Doob, 1953, Chapter VII, Theorem 2.3). Thus we obtain

$$\mathbb{P}(\text{equation (6) is true}) \leq \sum_{t=1}^L \sum_{u=1}^m \exp\left(-2u \frac{2 \log M}{u}\right) = LmM^{-4}.$$

The same reasoning gives

$$\mathbb{P}(\text{equation (7) is true}) \leq mhM^{-4},$$

which concludes the proof. ■

The next lemma proves that, if a sequence of actions has already been pulled enough, then equation (8) is not satisfied, and thus using lemmas 5 and 6 we deduce that with high probability this sequence of actions will not be selected anymore. This reasoning is made precise in Lemma 8.

Lemma 7 *Let $1 \leq h \leq L$, $a \in \mathcal{J}_h$ and $0 \leq h' < h$. Then equation (8) is not satisfied if the two following propositions are true:*

$$\forall 0 \leq t \leq h', T_{a_{1:t}}(m) \geq \frac{8}{\gamma^2} (h+1)^2 \gamma^{2(t-h)} \log M, \quad (10)$$

and

$$T_a(m) \geq \frac{8}{\gamma^2} (h+1)^2 \gamma^{2(h'-h)} \log M. \quad (11)$$

Proof: Assume that (10) and (11) are true. Then we clearly have:

$$\begin{aligned} 2 \sum_{t=1}^h \gamma^t \sqrt{\frac{2 \log M}{T_{a_{1:t}}(m)}} & = 2 \mathbf{1}_{h' > 0} \sum_{t=1}^{h'} \gamma^t \sqrt{\frac{2 \log M}{T_{a_{1:t}}(m)}} + 2 \sum_{t=h'+1}^h \gamma^t \sqrt{\frac{2 \log M}{T_{a_{1:t}}(m)}} \\ & \leq \frac{\gamma^{h+1}}{h+1} h' + \frac{\gamma^{h+1}}{h+1} \sum_{t=h'+1}^h \gamma^{t-h'} \\ & \leq \frac{\gamma^{h+1}}{h+1} \left(h' + \frac{\gamma}{1-\gamma} \right) \\ & \leq \frac{\gamma^{h+1}}{1-\gamma}, \end{aligned}$$

which proves the result. ■

Lemma 8 *Let $1 \leq h \leq L$, $a \in \mathcal{J}_h$ and $0 \leq h' < h$. Then $\tau_{h,h'}^a(m+1) = 1$ implies that either equation (6) or (7) is satisfied or the following proposition is true:*

$$\exists 0 \leq t \leq h' : |\mathcal{P}_{h,h'}^{a_{1:t}}(m)| < \gamma^{2(t-h')}. \quad (12)$$

Proof: If $\tau_{h,h'}^a(m+1) = 1$ then it means that $a^{m+1} \in aA^*$ and (11) is satisfied. By Lemma 5 this implies that either (6), (7) or (8) is true and (11) is satisfied. Now by Lemma 7 this implies that (6) is true or (7) is true or (10) is false. We now prove that if (12) is not satisfied then (10) is true, which clearly ends the proof. This follows from: For any $0 \leq t \leq h'$,

$$T_{a_{1:t}}(m) = \sum_{b \in a_{1:t}A^{h-t}} T_b(m) \geq \gamma^{2(t-h')} \frac{8}{\gamma^2} (h+1)^2 \gamma^{2(h'-h)} \log M = \frac{8}{\gamma^2} (h+1)^2 \gamma^{2(t-h)} \log M.$$

The next lemma is the key step of our proof. Intuitively, using lemmas 5 and 8, we have a good control on sequences for which equation (12) is satisfied. Note that (12) is a property which depends on sub-sequences of a from length 1 to h' . In the following proof we will iteratively "drop" all sequences which do not satisfy (12) from length t onwards, starting from $t = 1$. Then, on the remaining sequences, we can apply Lemma 8. ■

Lemma 9 *Let $1 \leq h \leq L$ and $0 \leq h' < h$. Then the following holds true,*

$$\mathbb{E}|\mathcal{P}_{h,h'}^\emptyset(M)| = \tilde{O} \left(\gamma^{-2h'} \mathbb{1}_{h' > 0} \sum_{t=0}^{h'} (\gamma^2 \kappa')^t + (\kappa')^h M^{-2} \right).$$

Proof: Let $h' \geq 1$ and $0 \leq s \leq h'$. We introduce the following random variables:

$$m_s^a = \min \left(M, \min \left\{ m \geq 0 : |\mathcal{P}_{h,h'}^a(m)| \geq \gamma^{2(s-h')} \right\} \right).$$

We will prove recursively that,

$$|\mathcal{P}_{h,h'}^\emptyset(m)| \leq \sum_{t=0}^s \gamma^{2(t-h')} |\mathcal{I}_t| + \sum_{a \in \mathcal{I}_s} \left| \mathcal{P}_{h,h'}^a \setminus \cup_{t=0}^s \mathcal{P}_{h,h'}^{a_{1:t}}(m_t^{a_{1:t}}) \right|. \quad (13)$$

The result is true for $s = 0$ since $\mathcal{I}_0 = \{\emptyset\}$ and by definition of m_0^\emptyset ,

$$|\mathcal{P}_{h,h'}^\emptyset(m)| \leq \gamma^{-2h'} + |\mathcal{P}_{h,h'}^\emptyset(m) \setminus \mathcal{P}_{h,h'}^\emptyset(m_0^\emptyset)|.$$

Now let us assume that the result is true for $s < h'$. We have:

$$\begin{aligned} \sum_{a \in \mathcal{I}_s} \left| \mathcal{P}_{h,h'}^a(m) \setminus \cup_{t=0}^s \mathcal{P}_{h,h'}^{a_{1:t}}(m_t^{a_{1:t}}) \right| &= \sum_{a \in \mathcal{I}_{s+1}} \left| \mathcal{P}_{h,h'}^a(m) \setminus \cup_{t=0}^s \mathcal{P}_{h,h'}^{a_{1:t}}(m_t^{a_{1:t}}) \right| \\ &\leq \sum_{a \in \mathcal{I}_{s+1}} \gamma^{2(s+1-h')} + \left| \mathcal{P}_{h,h'}^a(m) \setminus \cup_{t=0}^{s+1} \mathcal{P}_{h,h'}^{a_{1:t}}(m_t^{a_{1:t}}) \right| \\ &= \gamma^{2(s+1-h')} |\mathcal{I}_{s+1}| + \sum_{a \in \mathcal{I}_{s+1}} \left| \mathcal{P}_{h,h'}^a(m) \setminus \cup_{t=0}^{s+1} \mathcal{P}_{h,h'}^{a_{1:t}}(m_t^{a_{1:t}}) \right|, \end{aligned}$$

which ends the proof of (13). Thus we proved (by taking $s = h'$ and $m = M$):

$$\begin{aligned} |\mathcal{P}_{h,h'}^\emptyset(M)| &\leq \sum_{t=0}^{h'} \gamma^{2(t-h')} |\mathcal{I}_t| + \sum_{a \in \mathcal{I}_{h'}} \left| \mathcal{P}_{h,h'}^a(M) \setminus \cup_{t=0}^{s+1} \mathcal{P}_{h,h'}^{a_{1:t}}(m_t^{a_{1:t}}) \right| \\ &= \sum_{t=0}^{h'} \gamma^{2(t-h')} |\mathcal{I}_t| + \sum_{a \in \mathcal{J}_h} \left| \mathcal{P}_{h,h'}^a(M) \setminus \cup_{t=0}^{s+1} \mathcal{P}_{h,h'}^{a_{1:t}}(m_t^{a_{1:t}}) \right| \end{aligned}$$

Now, for any $a \in \mathcal{J}_h$, let $\tilde{m} = \max_{0 \leq t \leq h'} m_t^{a_{1:t}}$. Note that for $m \geq \tilde{m}$, equation (12) is not satisfied. Thus we have

$$\begin{aligned} \left| \mathcal{P}_{h,h'}^a \setminus \bigcup_{t=0}^{s+1} \mathcal{P}_{h,h'}^{a_{1:t}}(m_t^{a_{1:t}}) \right| &= \sum_{m=\tilde{m}}^{M-1} \tau_{h,h'}^a(m+1) = \sum_{m=0}^{M-1} \tau_{h,h'}^a(m+1) \mathbb{1}\{(12) \text{ is not satisfied}\} \\ &\leq \sum_{m=0}^{M-1} \tau_{h,h'}^a(m+1) \mathbb{1}\{(6) \text{ or } (7) \text{ is satisfied}\}. \end{aligned}$$

where the last inequality results from Lemma 8. Hence, we proved:

$$|\mathcal{P}_{h,h'}^\emptyset| \leq \sum_{t=0}^{h'} \gamma^{2(t-h')} |\mathcal{I}_t| + \sum_{m=0}^{M-1} \sum_{a \in \mathcal{J}_h} \mathbb{1}\{(6) \text{ or } (7) \text{ is satisfied}\}.$$

Taking the expectation, using (5) and applying Lemma 6 yield the claimed bound for $h' \geq 1$.

Now for $h' = 0$ we need a modified version of Lemma 8. Indeed in this case one can directly prove that $\tau_{h,0}^a(m+1) = 1$ implies that either equation (6) or (7) is satisfied (this follows from the fact that $\tau_{h,0}^a(m+1) = 1$ always imply that (10) is true for $h' = 0$). Thus we obtain:

$$|\mathcal{P}_{h,h'}^\emptyset| = \sum_{m=0}^{M-1} \sum_{a \in \mathcal{J}_h} \tau_{h,0}^a(m+1) \leq \sum_{m=0}^{M-1} \sum_{a \in \mathcal{J}_h} \mathbb{1}\{(6) \text{ or } (7) \text{ is satisfied}\}.$$

Taking the expectation and applying Lemma 6 yield the claimed bound for $h' = 0$ and ends the proof. \blacksquare

Lemma 10 *Let $1 \leq h \leq L$. The following holds true,*

$$\mathbb{E} \sum_{a \in \mathcal{J}_h} T_a(M) = \tilde{O} \left(\gamma^{-2h} \sum_{h'=1}^h (\gamma^2 \kappa')^{h'} + (\kappa')^h (1 + \gamma^{-2h} M^{-2}) \right).$$

Proof: We have the following computations:

$$\begin{aligned} \sum_{a \in \mathcal{J}_h} T_a(M) &= \sum_{a \in \mathcal{J}_h \setminus \mathcal{P}_{h,h-1}^\emptyset} T_a(M) + \sum_{h'=1}^{h-1} \sum_{a \in \mathcal{P}_{h,h'}^\emptyset \setminus \mathcal{P}_{h,h'-1}^\emptyset} T_a(M) + \sum_{a \in \mathcal{P}_{h,0}^\emptyset} T_a(M) \\ &\leq \frac{8}{\gamma^2} (h+1)^2 \gamma^{2(h-1-h)} |\mathcal{J}_h| + \sum_{h'=1}^{h-1} \frac{8}{\gamma^2} (h+1)^2 \gamma^{2(h'-1-h)} \log M |\mathcal{P}_{h,h'}^\emptyset| + M |\mathcal{P}_{h,0}^\emptyset| \\ &= \tilde{O} \left((\kappa')^h + \gamma^{-2h} \sum_{h'=1}^{h-1} \gamma^{2h'} |\mathcal{P}_{h,h'}^\emptyset| + M |\mathcal{P}_{h,0}^\emptyset| \right). \end{aligned}$$

Taking the expectation and applying the bound of Lemma 9 gives the claimed bound. \blacksquare

Thus by combining Lemma 4 and 10 we obtain for $\kappa' \gamma^2 \leq 1$:

$$\mathbb{E} r_n = \tilde{O} \left(\gamma^H + \gamma^{-H} M^{-1} + (\kappa')^H \gamma^{-H} M^{-3} \right),$$

and for $\kappa' \gamma^2 > 1$:

$$\mathbb{E} r_n = \tilde{O} \left(\gamma^H + (\kappa' \gamma)^H M^{-1} + (\kappa')^H \gamma^{-H} M^{-3} \right).$$

Thus in the case $\kappa' \gamma^2 \leq 1$, taking $H = \lceil \log M / (2 \log 1/\gamma) \rceil$ yields the claimed bound; while for $\kappa' \gamma^2 > 1$ we take $H = \lceil \log M / \log \kappa' \rceil$. Note that in both cases we have $H \leq L$ (as it was required at the beginning of the analysis).

Principal Component Analysis with Contaminated Data: The High Dimensional Case

Huan Xu

The University of Texas at Austin
huan.xu@mail.utexas.edu

Constantine Caramanis

The University of Texas at Austin
caramanis@mail.utexas.edu

Shie Mannor

Technion, Israel
shie@ee.technion.ac.il

Abstract

We consider the dimensionality-reduction problem (finding a subspace approximation of observed data) for contaminated data in the high dimensional regime, where the number of *observations* is of the same magnitude as the number of *variables* of each observation, and the data set contains some (arbitrarily) corrupted observations. We propose a High-dimensional Robust Principal Component Analysis (HR-PCA) algorithm that is tractable, robust to contaminated points, and easily kernelizable. The resulting subspace has a bounded deviation from the desired one, achieves maximal robustness – a breakdown point of 50% while all existing algorithms have a breakdown point of *zero*, and unlike ordinary PCA algorithms, achieves optimality in the limit case where the proportion of corrupted points goes to zero.

1 Introduction

The analysis of very high dimensional data – data sets where the dimensionality of each observation is comparable to or even larger than the number of observations – has drawn increasing attention in the last few decades (Donoho, 2000; Johnstone, 2001). For example, observations on individual instances can be curves, spectra, images or even movies, where a single observation has dimensionality ranging from thousands to billions. Practical high dimensional data examples include DNA Microarray data, financial data, climate data, web search engine, and consumer data. In addition, the nowadays standard “Kernel Trick” (Schölkopf & Smola, 2002) transforms virtually every data set to a high dimensional one. Efforts of extending traditional statistical tools (designed for the low dimensional case) into this high-dimensional regime are generally unsuccessful. This fact has stimulated research on formulating new data-analysis techniques able to cope with such a “dimensionality explosion.”

Principal Component Analysis (PCA) is one of the most widely used statistical techniques for dimensionality reduction. Work on PCA dates back as early as Pearson (1901), and has become one of the most important techniques for data compression and feature extraction. It is widely used in statistical data analysis, communication, pattern recognition, and image processing (Jolliffe, 1986). The standard PCA algorithm constructs the optimal (in a least-square sense) subspace approximation to observations by computing the eigenvectors or Principal Components (PCs) of the sample covariance or correlation matrix. Its broad application can be attributed to primarily two features: its success in the classical regime for recovering a low-dimensional subspace even in the presence of noise, and also the existence of efficient algorithms for computation. It is well-known, however, that precisely because of the quadratic error criterion, standard PCA is exceptionally fragile, and the quality of its output can suffer dramatically in the face of only a few (even a vanishingly small fraction) grossly corrupted points. Such non-probabilistic errors may be present due to data corruption stemming from sensor failures, malicious tampering, or other reasons. Attempts to use other error functions growing more slowly than the quadratic that might be more robust to outliers, results in non-convex (and intractable) problems.

In this paper, we consider a high-dimensional counterpart of Principal Component Analysis (PCA) that is robust to the existence of *arbitrarily corrupted* or contaminated data. We start with the standard statistical setup: a low dimensional signal is (linearly) mapped to a very high dimensional space, after which a high-dimensional Gaussian noise is added, to produce points that no longer lie on a low dimensional subspace. At this point, we deviate from the standard setting

in two important ways: (1) *a constant fraction of the points are arbitrarily corrupted* in a possibly non-probabilistic manner. We emphasize that these “outliers” can be entirely arbitrary, rather than from the tails of any particular distribution, e.g., the noise distribution; we call the remaining points “authentic”; (2) *the number of data points is of the same order as (or perhaps considerably smaller than) the dimensionality*. As we discuss below, these two points confound (to the best of our knowledge) all tractable existing robust PCA algorithms.

A fundamental feature of the high dimensionality is that the noise is large in some direction, with very high probability, and therefore standard definitions of “outliers” are of limited use in this setting. Another important property of this setup is that the signal-to-noise ratio (SNR) can go to zero, as the ℓ_2 norm of the high-dimensional Gaussian noise scales as the square root of the dimensionality. In the standard (i.e., low-dimensional case), a low SNR generally implies that the signal cannot be recovered, even without any corrupted points.

The Main Result

In this paper, we give a surprisingly optimistic message: contrary to what one might expect given the brittle nature of classical PCA, and in stark contrast to previous algorithms, it is possible to recover such low SNR signals, in the high-dimensional regime, even in the face of a *constant fraction of arbitrarily corrupted data*. Moreover, we show that this can be accomplished with an efficient (polynomial time) algorithm, which we call High-Dimensional Robust PCA (HR-PCA). The algorithm we propose here is tractable, provably robust to corrupted points, and asymptotically optimal, recovering the *exact* low-dimensional subspace when the number of corrupted points scales more slowly than the number of “authentic” samples – to the best of our knowledge, the only algorithm of this kind. Moreover, it is easily kernelizable.

Organization and Notation

In Section 2 we discuss past work and the reasons that classical robust PCA algorithms fail to extend to the high dimensional regime. In Section 3 we present the setup of the problem, and the HR-PCA algorithm. We also provide finite sample and asymptotic performance guarantees. The performance guarantees are proved in Section 4. Kernelization, simulation and some technical details in the derivation of the performance guarantees are postponed to the full version (Xu et al., 2010).

Capital letters and boldface letters are used to denote matrices and vectors, respectively. A $k \times k$ unit matrix is denoted by I_k . For $c \in \mathbb{R}$, $[c]^+ \triangleq \max(0, c)$. We let $\mathcal{B}_d \triangleq \{\mathbf{w} \in \mathbb{R}^d \mid \|\mathbf{w}\| \leq 1\}$, and \mathcal{S}_d be its boundary. We use a subscript (\cdot) to represent order statistics of a random variable. For example, let $v_1, \dots, v_n \in \mathbb{R}$. Then $v_{(1)}, \dots, v_{(n)}$ is a permutation of v_1, \dots, v_n , in a non-decreasing order.

2 Relation to Past Work

In this section, we discuss past work and the reasons that classical robust PCA algorithms fail to extend to the high dimensional regime.

Much previous robust PCA work focuses on the traditional robustness measurement known as the “breakdown point” (Huber, 1981), i.e., the percentage of corrupted points that can make the output of the algorithm *arbitrarily* bad. To the best of our knowledge, no other algorithm can handle *any constant fraction of outliers* with a lower bound on the error in the high-dimensional regime. That is, the best-known breakdown point for this problem is zero. We show that the algorithm we provide has breakdown point of 50%, which is the best break-down point possible for any algorithm. In addition to this, we focus on providing explicit lower bounds on the performance, for all corruption levels up to the breakdown point.

In the low-dimensional regime where the observations significantly outnumber the variables of each observation, several robust PCA algorithms have been proposed (e.g., Devlin et al., 1981; Xu & Yuille, 1995; Yang & Wang, 1999; Croux & Hasebroeck, 2000; De la Torre & Black, 2001; De la Torre & Black, 2003; Croux et al., 2007; Brubaker, 2009).

We discuss three main pitfalls these and other existing algorithms face in high dimensions.

Diminishing Breakdown Point: If an algorithm’s breakdown point has an inverse dependence on the dimensionality, then it is unsuitable in our regime. Many algorithms fall into this category. In Donoho (1982), several covariance estimators including M-estimator (Maronna, 1976), Convex Peeling (Barnett, 1976; Bebbington, 1978), Ellipsoidal Peeling (Titterton, 1978; Helbling, 1983), Classical Outlier Rejection (Barnett & Lewis, 1978; David, 1981), Iterative Deletion (Dempster & Gasko-Green, 1981) and Iterative Trimming (Gnanadesikan & Kettenring, 1972; Devlin et al., 1975) are all shown to have breakdown points upper-bounded by the inverse of the dimensionality, hence not useful in the regime of interest.

Noise Explosion: In the basic PCA model, zero mean standard Gaussian noise is added to each sample observed. Concentration results for Gaussian vectors promise that the noise magnitude will sharply concentrate around the ball of radius equal to square root of the dimension. This can be significantly larger than what we call the “signal strength,” namely, the magnitude of the signal before noise was added. Thus, the ratio of the signal strength to the noise level quickly goes to zero as we scale the dimensionality up. Because of this, several perhaps counter-intuitive properties hold in this regime. First, any given authentic point is with overwhelming probability very close to orthogonal to the signal space (i.e., to the true principal components). Second, it is possible for a constant fraction of corrupted points all with a small Mahalanobis distance to significantly change the output of PCA. Indeed, by aligning the entire fraction of corrupted points magnitude some constant multiple of what we have called the signal strength, it is easy to see that the output of PCA can be strongly manipulated. On the other hand, since the noise magnitude is much larger, and in a direction perpendicular to the principal components, the Mahalanobis distance of each corrupted point will be very small. Third, the same example as above shows that it is possible for a constant fraction of corrupted points all with small Stahel-Donoho (S-D) outlyingness to significantly change the output of PCA, where recall that S-D outlyingness of a sample \mathbf{y}_i is defined as:

$$u_i \triangleq \sup_{\|\mathbf{w}\|=1} \frac{|\mathbf{w}^\top \mathbf{y}_i - \text{med}_j(\mathbf{w}^\top \mathbf{y}_j)|}{\text{med}_k |\mathbf{w}^\top \mathbf{y}_k - \text{med}_j(\mathbf{w}^\top \mathbf{y}_j)|}.$$

Here med_k stands for taking median over all k .

The Mahalanobis distance and the S-D outlyingness are extensively used in existing robust PCA algorithms. For example, Classical Outlier Rejection, Iterative Deletion and various alternatives of Iterative Trimmings all use the Mahalanobis distance to identify possible outliers. Depth Trimming (Donoho, 1982) weights the contribution of observations based on their S-D outlyingness. More recently, the ROBPCA algorithm proposed in Hubert et al. (2005) selects a subset of observations with least S-D outlyingness to compute the d -dimensional signal space. Thus, in the high-dimensional case, these algorithms may run into problems since neither Mahalanobis distance nor S-D outlyingness are valid indicators of outliers. Indeed, as shown in the simulations, the empirical performance of such algorithms can be worse than standard PCA, because they remove the authentic samples.

Algorithmic Tractability: There are algorithms that do not rely on Mahalanobis distance or S-D outlyingness, and have a non-diminishing breakdown point, namely Minimum Volume Ellipsoid (MVE), Minimum Covariance Determinant (MCD) (Rousseeuw, 1984) and Projection-Pursuit (Li & Chen, 1985). MVE finds the minimum volume ellipsoid that covers a certain fraction of observations. MCD finds a fraction of observations whose covariance matrix has a minimal determinant. Projection Pursuit maximizes a certain robust univariate variance estimator over all directions.

MCD and MVE are combinatorial, and hence (as far as we know) computationally intractable as the size of the problem scales. More difficult yet, MCD and MVE are ill-posed in the high-dimensional setting where the number of points (roughly) equals the dimension, since there exist infinitely many zero-volume (determinant) ellipsoids satisfying the covering requirement. Nevertheless, we note that such algorithms work well in the low-dimensional case, and hence can potentially be used as a post-processing procedure of our algorithm by projecting all observations to the output subspace to fine tune the eigenvalues and eigenvectors we produce.

Maximizing a robust univariate variance estimator as in Projection Pursuit, is also non-convex, and thus to the best of our knowledge, computationally intractable. In Croux and Ruiz-Gazen (2005), the authors propose a fast Projection-Pursuit algorithm, avoiding the non-convex optimization problem of finding the optimal direction, by only examining the directions of each sample. While this is suitable in the classical regime, in the high-dimensional setting this algorithm fails, since as discussed above, the direction of each sample is almost orthogonal to the direction of true principal components. Such an approach would therefore only be examining candidate directions nearly orthogonal to the true maximizing directions.

Low Rank Techniques: Finally, we discuss the recent (as of yet unpublished) paper (Candès et al., 2009). In this work, the authors adapt techniques from low-rank matrix approximation, and in particular, results similar to the matrix decomposition results of Chandrasekaran et al. (2009), in order to recover a low-rank matrix L_0 from highly corrupted measurements $M = L_0 + S_0$, where the noise term, S_0 , is assumed to have a sparse structure. This models the scenario where we have perfect measurement of most of the entries of L_0 , and a small (but constant) fraction of the entries are arbitrarily corrupted. This work is much closer in spirit, in motivation, and in terms of techniques, to the low-rank matrix completion and matrix recovery problems in Candès and Recht (2009); Recht (2009); Recht et al. (2010) than the setting we consider and the work presented herein. In particular, in our setting, even one corrupted point can change *every* element of the measurement

M .

3 HR-PCA: The Algorithm

The algorithm of HR-PCA is presented in this section. We start with the mathematical setup of the problem in Section 3.1. The HR-PCA algorithm as well as its performance guarantee are then given in Section 3.2.

3.1 Problem Setup

We now define in detail the problem described above.

- The “authentic samples” $\mathbf{z}_1, \dots, \mathbf{z}_t \in \mathbb{R}^m$ are generated by $\mathbf{z}_i = A\mathbf{x}_i + \mathbf{n}_i$, where $\mathbf{x}_i \in \mathbb{R}^d$ (the “signal”) are i.i.d. samples of a random variable \mathbf{x} , and \mathbf{n}_i (the “noise”) are independent realizations of $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, I_m)$. The matrix $A \in \mathbb{R}^{m \times d}$ and the distribution of \mathbf{x} (denoted by μ) are unknown. We do assume, however, that the distribution μ is absolutely continuous with respect to the Borel measure, it is spherically symmetric (and in particular, \mathbf{x} has mean zero and variance I_d) and it has light tails, specifically, there exist constants K and $C > 0$ such that $\Pr(\|\mathbf{x}\| \geq x) \leq K \exp(-Cx)$ for all $x \geq 0$. Since the distribution μ and the dimension d are both fixed, as m, n scale, the assumption that μ is spherically symmetric can be easily relaxed, and the expense of potentially significant *notational complexity*.
- The outliers (the corrupted data) are denoted $\mathbf{o}_1, \dots, \mathbf{o}_{n-t} \in \mathbb{R}^m$ and as emphasized above, they are arbitrary (perhaps even maliciously chosen). We denote the fraction of corrupted points by $\lambda \triangleq (n-t)/n$.
- We only observe the contaminated data set

$$\mathcal{Y} \triangleq \{\mathbf{y}_1, \dots, \mathbf{y}_n\} = \{\mathbf{z}_1, \dots, \mathbf{z}_t\} \cup \{\mathbf{o}_1, \dots, \mathbf{o}_{n-t}\}.$$

An element of \mathcal{Y} is called a “point”.

Given these contaminated observations, we want to recover the principal components of A , i.e., the top eigenvectors, $\bar{\mathbf{w}}_1, \dots, \bar{\mathbf{w}}_d$ of AA^\top . That is, we seek a collection of orthogonal vectors $\mathbf{w}_1, \dots, \mathbf{w}_d$, that maximize the performance metric called the *Expressed Variance* (E.V.):

$$\text{E.V.}(\mathbf{w}_1, \dots, \mathbf{w}_d) \triangleq \frac{\sum_{j=1}^d \mathbf{w}_j^\top AA^\top \mathbf{w}_j}{\sum_{j=1}^d \bar{\mathbf{w}}_j^\top AA^\top \bar{\mathbf{w}}_j} = \frac{\sum_{j=1}^d \mathbf{w}_j^\top AA^\top \mathbf{w}_j}{\text{trace}(AA^\top)}.$$

The E.V. is always less than one, with equality achieved exactly when the vectors $\mathbf{w}_1, \dots, \mathbf{w}_d$ have the span of the true principal components $\{\bar{\mathbf{w}}_1, \dots, \bar{\mathbf{w}}_d\}$. When $d = 1$, the Expressed Variance relates to another natural performance metric — the angle between \mathbf{w}_1 and $\bar{\mathbf{w}}_1$ — since by definition $\text{E.V.}(\mathbf{w}_1) = \cos^2(\angle(\mathbf{w}_1, \bar{\mathbf{w}}_1))$.¹ The Expressed Variance represents the portion of signal $A\mathbf{x}$ being expressed by $\mathbf{w}_1, \dots, \mathbf{w}_d$. Equivalently, $1 - \text{E.V.}$ is the reconstruction error of the signal.

It is natural to expect that the ability to recover vectors with a high expressed variance depends on λ , the fraction of corrupted points — in addition, it depends on the distribution, μ generating the (low-dimensional) points \mathbf{x} , through its tails. If μ has longer tails, outliers that affect the variance (and hence are far from the origin) and authentic samples in the tail of the distribution, become more difficult to distinguish. To quantify this effect, we define the following “tail weight” function $\mathcal{V} : [0, 1] \rightarrow [0, 1]$:

$$\mathcal{V}(\alpha) \triangleq \int_{-c_\alpha}^{c_\alpha} x^2 \bar{\mu}(dx);$$

where $\bar{\mu}$ is the one-dimensional margin of μ (recall that μ is spherically symmetric), and c_α is such that $\bar{\mu}([-c_\alpha, c_\alpha]) = \alpha$. Since μ has a density function, c_α is well defined. Thus, $\mathcal{V}(\cdot)$ represents how the tail of $\bar{\mu}$ contributes to its variance. Notice that $\mathcal{V}(0) = 0$, $\mathcal{V}(1) = 1$, and $\mathcal{V}(\cdot)$ is continuous in $[0, 1]$ since μ has a density function. For notational convenience, we simply let $\mathcal{V}(x) = 0$ for $x < 0$, and $\mathcal{V}(x) = \infty$ for $x > 1$.

The bounds on the quality of recovery, given in Theorems 1 and 2 below, are functions of η and the function $\mathcal{V}(\cdot)$.

¹This geometric interpretation does not extend to the case where $d > 1$, since the angle between two subspaces is not well defined.

High Dimensional Setting and Asymptotic Scaling

In this paper, we focus on the case where $n \sim m \gg d$ and $\text{trace}(A^\top A) \gg 1$. That is, the number of observations and the dimensionality are of the same magnitude, and much larger than the dimensionality of \mathbf{x} ; the trace of $A^\top A$ is significantly larger than 1, but may be much smaller than n and m . In our asymptotic scaling, n and m scale together to infinity, while d remains fixed. The value of $\text{trace}(A^\top A)$ also scales to infinity, but there is no lower bound on the rate at which this happens (and in particular, the scaling of $\text{trace}(A^\top A)$ can be much slower than the scaling of m and n).

While we give finite-sample results, we are particularly interested in the asymptotic performance of HR-PCA when *the dimension and the number of observations grow together* to infinity. Our asymptotic setting is as follows. Suppose there exists a sequence of sample sets $\{\mathcal{Y}(j)\} = \{\mathcal{Y}(1), \mathcal{Y}(2), \dots\}$, where for $\mathcal{Y}(j)$, $n(j)$, $m(j)$, $A(j)$, $d(j)$, etc., denote the corresponding values of the quantities defined above. Then the following must hold for some positive constants c_1, c_2 :

$$\begin{aligned} \lim_{j \rightarrow \infty} \frac{n(j)}{m(j)} &= c_1; & d(j) &\leq c_2; & m(j) &\uparrow +\infty; \\ \text{trace}(A(j)^\top A(j)) &\uparrow +\infty. \end{aligned} \tag{1}$$

While $\text{trace}(A(j)^\top A(j)) \uparrow +\infty$, if it scales more slowly than $\sqrt{m(j)}$, the SNR will asymptotically decrease to zero.

3.2 Key Idea and Main Algorithm

For $\mathbf{w} \in \mathcal{S}_m$, we define the Robust Variance Estimator (RVE) as $\bar{V}_{\hat{t}}(\mathbf{w}) \triangleq \frac{1}{n} \sum_{i=1}^{\hat{t}} |\mathbf{w}^\top \mathbf{y}_{(i)}|^2$. This stands for the following statistics: project \mathbf{y}_i onto the direction \mathbf{w} , replace the furthest (from original) $n - \hat{t}$ samples by 0, and then compute the variance. Notice that the RVE is always performed on the original observed set \mathcal{Y} .

The main algorithm of HR-PCA is as given below.

Algorithm 1 HR-PCA

Input: Contaminated sample-set $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\} \subset \mathbb{R}^m$, d, \bar{T}, \hat{t} .

Output: $\mathbf{w}_1^*, \dots, \mathbf{w}_d^*$.

Algorithm:

1. Let $\hat{\mathbf{y}}_i := \mathbf{y}_i$ for $i = 1, \dots, n$; $s := 0$; Opt := 0.
2. While $s \leq \bar{T}$, do
 - (a) Compute the empirical variance matrix

$$\hat{\Sigma} := \frac{1}{n-s} \sum_{i=1}^{n-s} \hat{\mathbf{y}}_i \hat{\mathbf{y}}_i^\top.$$

- (b) Perform PCA on $\hat{\Sigma}$. Let $\mathbf{w}_1, \dots, \mathbf{w}_d$ be the d principal components of $\hat{\Sigma}$.
- (c) If $\sum_{j=1}^d \bar{V}_{\hat{t}}(\mathbf{w}_j) > \text{Opt}$, then let $\text{Opt} := \sum_{j=1}^d \bar{V}_{\hat{t}}(\mathbf{w}_j)$ and let $\mathbf{w}_j^* := \mathbf{w}_j$ for $j = 1, \dots, d$.
- (d) Randomly remove a point from $\{\hat{\mathbf{y}}_i\}_{i=1}^{n-s}$ according to

$$\Pr(\hat{\mathbf{y}}_i \text{ is removed}) \propto \sum_{j=1}^d (\mathbf{w}_j^\top \hat{\mathbf{y}}_i)^2;$$

- (e) Denote the remaining points by $\{\hat{\mathbf{y}}_i\}_{i=1}^{n-s-1}$;
 - (f) $s := s + 1$.
3. Output $\mathbf{w}_1^*, \dots, \mathbf{w}_d^*$. End.

Intuition on Why The Algorithm Works

On any given iteration, we select candidate directions based on standard PCA – thus directions chosen are those with largest empirical variance. Now, given a candidate direction, \mathbf{w} , our robust variance estimator measures the variance of the $(n - \hat{t})$ -smallest points projected in that direction. If this is large, it means that many of the points have a large variance in this direction – the points contributing to the robust variance estimator, and the points that led to this direction being selected

by PCA. If the robust variance estimator is small, it is likely that a number of the largest variance points are corrupted, and thus removing one of them randomly, in proportion to their distance in the direction \mathbf{w} , will remove a corrupted point.

Thus in summary, the algorithm works for the following intuitive reason. If the corrupted points have a very high variance along a direction with large angle from the span of the principal components, then with some probability, our algorithm removes them. If they have a high variance in a direction “close to” the span of the principal components, then this can only help in finding the principal components. Finally, if the corrupted points do not have a large variance, then the distortion they can cause in the output of PCA is necessarily limited.

The remainder of the paper makes this intuition precise, providing lower bounds on the probability of removing corrupted points, and subsequently upper bounds on the maximum distortion the corrupted points can cause, i.e., lower bounds on the expressed variance of the principal components the algorithm recovers.

There are two parameters to tune for HR-PCA, namely \hat{t} and \bar{T} . Basically, \hat{t} affects the performance of HR-PCA through Inequality 2, and as a rule of thumb we can set $\hat{t} = t$ if no *a priori* information of μ exists. (Note that our algorithm does assume knowledge of at least a lower bound on the number of authentic points, or, equivalently, an upper bound on λ , the fraction of corrupted points.) \bar{T} does not affect the performance as long as it is large enough, hence we can simply set $T = n - 1$, although when λ is small, a smaller T leads to the same solution with less computational cost.

The correctness of HR-PCA is shown in the following theorems for both the finite-sample bound, and the asymptotic performance.

Theorem 1 (Finite Sample Performance) *Let the algorithm above output $\{\mathbf{w}_1, \dots, \mathbf{w}_d\}$. Fix a $\kappa > 0$, and let $\tau = \max(m/n, 1)$. There exists a universal constant c_0 and a constant C which can possibly depend on \hat{t}/t , λ , d , μ and κ , such that for any $\gamma < 1$, if $n/\log^4 n \geq \log^6(1/\gamma)$, then with probability $1 - \gamma$ the following holds*

$$\begin{aligned} \text{E.V.}\{\mathbf{w}_1, \dots, \mathbf{w}_d\} &\geq \left[\frac{\mathcal{V}\left(1 - \frac{\lambda(1+\kappa)}{(1-\lambda)\kappa}\right)}{(1+\kappa)} \right] \times \left[\frac{\mathcal{V}\left(\frac{\hat{t}}{t} - \frac{\lambda}{1-\lambda}\right)}{\mathcal{V}\left(\frac{\hat{t}}{t}\right)} \right] \\ &\quad - \left[\frac{8\sqrt{c_0\tau d}}{\mathcal{V}\left(\frac{\hat{t}}{t}\right)} \right] (\text{trace}(AA^\top))^{-1/2} - \left[\frac{2c_0\tau}{\mathcal{V}\left(\frac{\hat{t}}{t}\right)} \right] (\text{trace}(AA^\top))^{-1} - C \frac{\log^2 n \log^3(1/\gamma)}{\sqrt{n}}. \end{aligned}$$

The last three terms go to zero as the dimension and number of points scale to infinity, i.e., as n and $m \rightarrow \infty$. Therefore, we immediately obtain:

Theorem 2 (Asymptotic Performance) *Given a sequence of $\{\mathcal{Y}(j)\}$, if the asymptotic scaling in Expression (1) holds, and $\limsup \lambda(j) \leq \lambda^*$, then the following holds in probability when $j \uparrow \infty$ (i.e., when n and $m \uparrow \infty$),*

$$\liminf_j \text{E.V.}\{\mathbf{w}_1(j), \dots, \mathbf{w}_d(j)\} \geq \max_\kappa \left[\frac{\mathcal{V}\left(1 - \frac{\lambda^*(1+\kappa)}{(1-\lambda^*)\kappa}\right)}{(1+\kappa)} \right] \times \left[\frac{\mathcal{V}\left(\frac{\hat{t}}{t} - \frac{\lambda^*}{1-\lambda^*}\right)}{\mathcal{V}\left(\frac{\hat{t}}{t}\right)} \right]. \quad (2)$$

Remark

1. The bounds in the two bracketed terms in the asymptotic bound may be, roughly, explained as follows. The first term is due to the fact that the removal procedure may well not remove all large-magnitude corrupted points, while at the same time, some authentic points may be removed. The second term accounts for the fact that not all the outliers may have large magnitude. These will likely not be removed, and will have some (small) effect on the principal component directions reported in the output.
2. The terms in the second line of Theorem 1 go to zero as n and m increases, and therefore Theorem 1 immediately implies Theorem 2.
3. If $\lambda(j) \downarrow 0$, i.e., the number of corrupted points scales sublinearly (in particular, this holds when there are a fixed number of corrupted points), then the right-hand-side of Inequality (2) equals

1,² i.e., HR-PCA is asymptotically optimal. This is in contrast to PCA, where the existence of *even a single* corrupted point is sufficient to bound the output *arbitrarily* away from the optimum.

4. The breakdown point of HR-PCA converges to 50%. Note that since μ has a density function, $\mathcal{V}(\alpha) > 0$ for any $\alpha \in (0, 1]$. Therefore, for any $\lambda < 1/2$, if we set \hat{t} to any value in $(\lambda n, t]$, then there exists κ large enough such that the right-hand-side is strictly positive (recall that $t = (1 - \lambda)n$). The breakdown point hence converges to 50%. Thus, HR-PCA achieves the maximal possible break-down point (note that a breakdown point greater than 50% is never possible, since then there are more outliers than samples).

The graphs in Figure 1 illustrate the lower-bounds of asymptotic performance if the 1-dimension marginal of μ is the Gaussian distribution or the Uniform distribution.

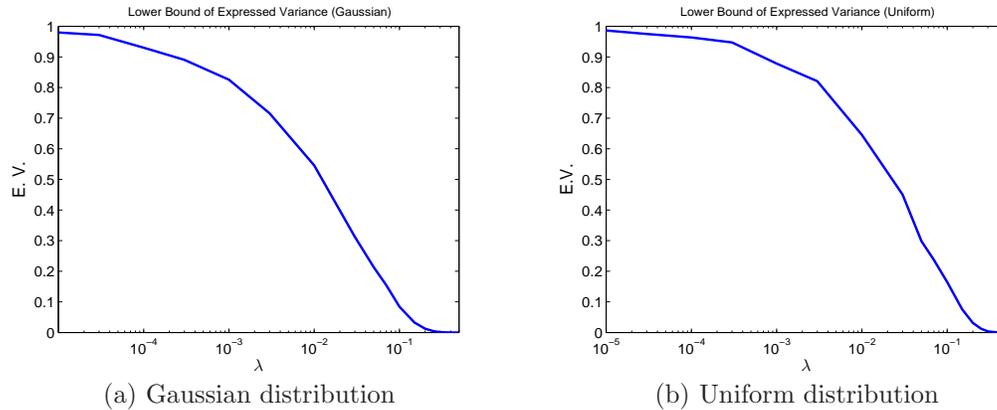


Figure 1: Lower Bounds of Asymptotic Performance.

We briefly discuss kernelizing HR-PCA : given a feature mapping $\Upsilon(\cdot) : \mathbb{R}^m \rightarrow \mathcal{H}$ equipped with a kernel function $k(\cdot, \cdot)$, we perform the dimensionality reduction in the feature space \mathcal{H} without knowing the explicit form of $\Upsilon(\cdot)$. Notice that HR-PCA involves finding a set of PCs $\mathbf{w}_1, \dots, \mathbf{w}_d \in \mathcal{H}$, and evaluating $\langle \mathbf{w}_q, \Upsilon(\cdot) \rangle$ (The RVE is a function of $\langle \mathbf{w}_q, \Upsilon(\mathbf{y}_i) \rangle$, and random removal depends on $\langle \mathbf{w}_q, \Upsilon(\hat{\mathbf{y}}_i) \rangle$). The former can be kernelized by applying the Kernel PCA algorithm introduced by Schölkopf et al. (1999), where each of the output PCs admits a representation $\mathbf{w}_q = \sum_{j=1}^{n-s} \alpha_j(q) \Upsilon(\hat{\mathbf{y}}_j)$. Thus, $\langle \mathbf{w}_q, \Upsilon(\cdot) \rangle$ is easily evaluated by $\langle \mathbf{w}_q, \Upsilon(\mathbf{v}) \rangle = \sum_{j=1}^{n-s} \alpha_j(q) k(\hat{\mathbf{y}}_j, \mathbf{v})$, for all $\mathbf{v} \in \mathbb{R}^m$, implying that HR-PCA can be kernelized. We leave the details to the full version (Xu et al., 2010). Due to space constraints, numerical simulations are also deferred to the full version (Xu et al., 2010).

4 Proof of the Main Result

In this section we provide the main steps of the proof of the finite-sample and asymptotic performance bounds, including the precise statements and the key ideas in the proof, but deferring some of the more standard or tedious elements to the full version (Xu et al., 2010). The proof consists of three steps which we now outline. In what follows, we let d , m/n , λ , \hat{t}/t , and μ be fixed. We can fix a $\lambda \in (0, 0.5)$ without loss of generality, due to the fact that if a result is shown to hold for λ , then it holds for $\lambda' < \lambda$. The letter c is used to represent a constant, and ϵ is a constant that decreases to zero as n and m increase to infinity. The values of c and ϵ can change from line to line, and can possibly depend on d , m/n , λ , \hat{t}/t , and μ .

1. The blessing of dimensionality, and laws of large numbers: The first step involves two ideas; the first is the well-known fact (e.g., Davidson & Szarek, 2001) as n and m scale, the expectation of the covariance of the noise is bounded *independently* of m . The second involves appealing to laws of large numbers to show that sample estimates of the covariance of the noise, \mathbf{n} , of the signal, \mathbf{x} , and then of the authentic points, $\mathbf{z} = \mathbf{A}\mathbf{x} + \mathbf{n}$, are uniformly close to their expectation, with high probability. Specifically, we prove that:

²We can take $\kappa(j) = \sqrt{\lambda(j)}$ and note that since μ has a density, $\mathcal{V}(\cdot)$ is continuous.

- (a) With high probability, the largest eigenvalue of the variance of noise matrix is bounded. That is,

$$\sup_{\mathbf{w} \in \mathcal{S}_m} \frac{1}{n} \sum_{i=1}^t (\mathbf{w}^\top \mathbf{n}_i)^2 \leq c.$$

- (b) With high probability, both the largest and the smallest eigenvalue of the signals in the original space converge to 1. That is

$$\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^t (\mathbf{w}^\top \mathbf{x}_i)^2 - 1 \right| \leq \epsilon.$$

- (c) Under 1b, with high probability, RVE is a valid variance estimator for the d -dimensional signals. That is,

$$\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^{\hat{t}} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V} \left(\frac{\hat{t}}{t} \right) \right| \leq \epsilon.$$

- (d) Under 1a and 1c, RVE is a valid estimator of the variance of the authentic samples. That is, the following holds uniformly over all $\mathbf{w} \in \mathcal{S}_m$,

$$(1 - \epsilon) \|\mathbf{w}^\top A\|^2 \mathcal{V} \left(\frac{t'}{t} \right) - c \|\mathbf{w}^\top A\| \leq \frac{1}{t} \sum_{i=1}^{t'} |\mathbf{w}^\top \mathbf{z}_{(i)}|^2 \leq (1 + \epsilon) \|\mathbf{w}^\top A\|^2 \mathcal{V} \left(\frac{t'}{t} \right) + c \|\mathbf{w}^\top A\|.$$

2. The next step shows that with high probability, the algorithm finds a “good” solution within a bounded number of steps. In particular, this involves showing that if in a given step the algorithm has not found a good solution, in the sense that the variance along a principal component is not mainly due to the authentic points, then the random removal scheme removes a corrupted point with probability bounded away from zero. We then use martingale arguments to show that as a consequence of this, there cannot be many steps with the algorithm finding at least one “good” solution, since in the absence of good solutions, most of the corrupted points are removed by the algorithm.
3. The previous step shows the existence of a “good” solution. The final step shows two things: first, that this good solution has performance that is close to that of the optimal solution, and second, that the final output of the algorithm is close to that of the “good” solution. Combining these two steps, we derive the finite-sample and asymptotic performance bounds for HR-PCA.

4.1 Step 1

We state the main results for Step 1a and 1b. The proofs are deferred to the full version (Xu et al., 2010). In a nutshell, they hold by applying Theorem II.13 of Davidson and Szarek (2001), and Theorem 2.1 of Mendelson and Pajor (2006), respectively.

Theorem 3 *There exist universal constants c and c' such that for any $\gamma > 0$, with probability at least $1 - \gamma$, the following holds:*

$$\sup_{\mathbf{w} \in \mathcal{S}_m} \frac{1}{t} \sum_{i=1}^t (\mathbf{w}^\top \mathbf{n}_i)^2 \leq c + \frac{c' \log \frac{1}{\gamma}}{n}.$$

Theorem 4 *There exists a constant c that only depends on μ and d , such that for any $\gamma > 0$, with probability at least $1 - \gamma$,*

$$\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^t (\mathbf{w}^\top \mathbf{x}_i)^2 - 1 \right| \leq \frac{c \log^2 n \log^3 \frac{1}{\gamma}}{\sqrt{n}}.$$

The next theorem is the main result for Step 1c. Briefly speaking, since d is fixed, the result holds due to a standard uniform convergence argument. See Xu et al. (2010) for details.

Theorem 5 *Fix $\eta < 1$. There exists a constant c that depends on d , μ and η , such that for all $\gamma < 1$, t , the following holds with probability at least $1 - \gamma$:*

$$\sup_{\mathbf{w} \in \mathcal{S}_d, \bar{t} \leq \eta t} \left| \frac{1}{t} \sum_{i=1}^{\bar{t}} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V} \left(\frac{\bar{t}}{t} \right) \right| \leq c \sqrt{\frac{\log n + \log 1/\gamma}{n}} + c \frac{\log^{5/2} n \log^{7/2}(1/\gamma)}{n}.$$

Recall that $\mathbf{z}_i = \mathbf{A}\mathbf{x}_i + \mathbf{n}_i$. Algebraic manipulation yields Theorem 6, which is the main result of Step 1d, and Corollary 7.

Theorem 6 *Let $t' \leq t$. If there exists $\epsilon_1, \epsilon_2, \bar{c}$ such that (I) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t'} \sum_{i=1}^{t'} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V}(\frac{t'}{t}) \right| \leq \epsilon_1$; (II) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{x}_i|^2 - 1 \right| \leq \epsilon_2$; (III) $\sup_{\mathbf{w} \in \mathcal{S}_m} \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{n}_i|^2 \leq \bar{c}$, then for all $\mathbf{w} \in \mathcal{S}_m$ the following holds:*

$$\begin{aligned} & (1 - \epsilon_1) \|\mathbf{w}^\top \mathbf{A}\|^2 \mathcal{V}\left(\frac{t'}{t}\right) - 2 \|\mathbf{w}^\top \mathbf{A}\| \sqrt{(1 + \epsilon_2)\bar{c}} \\ & \leq \frac{1}{t} \sum_{i=1}^{t'} |\mathbf{w}^\top \mathbf{z}_{(i)}|^2 \leq (1 + \epsilon_1) \|\mathbf{w}^\top \mathbf{A}\|^2 \mathcal{V}\left(\frac{t'}{t}\right) + 2 \|\mathbf{w}^\top \mathbf{A}\| \sqrt{(1 + \epsilon_2)\bar{c}} + \bar{c}. \end{aligned}$$

Corollary 7 *Let $t' \leq t$. If there exists $\epsilon_1, \epsilon_2, \bar{c}$ such that (I) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t'} \sum_{i=1}^{t'} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V}(\frac{t'}{t}) \right| \leq \epsilon_1$; (II) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{x}_i|^2 - 1 \right| \leq \epsilon_2$; (III) $\sup_{\mathbf{w} \in \mathcal{S}_m} \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{n}_i|^2 \leq \bar{c}$, then for any $\mathbf{w}_1, \dots, \mathbf{w}_d \in \mathcal{S}_m$, and let $H(\mathbf{w}_1, \dots, \mathbf{w}_d) \triangleq \sum_{j=1}^d \|\mathbf{w}_j^\top \mathbf{A}\|^2$, the following holds*

$$\begin{aligned} & (1 - \epsilon_1) \mathcal{V}\left(\frac{t'}{t}\right) H(\mathbf{w}_1, \dots, \mathbf{w}_d) - 2 \sqrt{(1 + \epsilon_2)\bar{c}dH(\mathbf{w}_1, \dots, \mathbf{w}_d)} \\ & \leq \sum_{j=1}^d \frac{1}{t} \sum_{i=1}^{t'} |\mathbf{w}_j^\top \mathbf{z}_{(i)}|^2 \leq (1 + \epsilon_1) \mathcal{V}\left(\frac{t'}{t}\right) H(\mathbf{w}_1, \dots, \mathbf{w}_d) + 2 \sqrt{(1 + \epsilon_2)\bar{c}dH(\mathbf{w}_1, \dots, \mathbf{w}_d)} + \bar{c}. \end{aligned}$$

Letting $t' = t$ we immediately have the following corollary.

Corollary 8 *If there exists ϵ, \bar{c} such that (I) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{x}_i|^2 - 1 \right| \leq \epsilon$; and (II) $\sup_{\mathbf{w} \in \mathcal{S}_m} \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{n}_i|^2 \leq \bar{c}$, then for any $\mathbf{w}_1, \dots, \mathbf{w}_d \in \mathcal{S}_m$ the following holds:*

$$\begin{aligned} & (1 - \epsilon) H(\mathbf{w}_1, \dots, \mathbf{w}_d) - 2 \sqrt{(1 + \epsilon)\bar{c}dH(\mathbf{w}_1, \dots, \mathbf{w}_d)} \\ & \leq \sum_{j=1}^d \frac{1}{t} \sum_{i=1}^t |\mathbf{w}_j^\top \mathbf{z}_i|^2 \leq (1 + \epsilon) H(\mathbf{w}_1, \dots, \mathbf{w}_d) + 2 \sqrt{(1 + \epsilon)\bar{c}dH(\mathbf{w}_1, \dots, \mathbf{w}_d)} + \bar{c}. \end{aligned}$$

4.2 Step 2

The next step shows that the algorithm finds a good solution in a small number of steps. Proving this involves showing that at any given step, either the algorithm finds a good solution, or the random removal eliminates one of the corrupted points with high probability (i.e., probability bounded away from zero). The intuition then, is that there cannot be too many steps without finding a good solution, since too many of the corrupted points will have been removed. This section makes this intuition precise.

Let us fix a $\kappa > 0$. Let $\mathcal{Z}(s)$ and $\mathcal{O}(s)$ be the set of remaining authentic samples and the set of remaining corrupted points after the s^{th} stage, respectively. Then with this notation, $\mathcal{Y}(s) = \mathcal{Z}(s) \cup \mathcal{O}(s)$. Observe that $|\mathcal{Y}(s)| = n - s$. Let $\bar{\mathbf{r}}(s) = \mathcal{Y}(s-1) \setminus \mathcal{Y}(s)$, i.e., the point removed at stage s . Let $\mathbf{w}_1(s), \dots, \mathbf{w}_d(s)$ be the d PCs found in the s^{th} stage — these points are the output of standard PCA on $\mathcal{Y}(s-1)$. These points are a good solution if the variance of the points projected onto their span is mainly due to the authentic samples rather than the corrupted points. We denote this “good output event at step s ” by $\mathcal{E}(s)$, defined as follows:

$$\mathcal{E}(s) = \left\{ \sum_{j=1}^d \sum_{\mathbf{z}_i \in \mathcal{Z}(s-1)} (\mathbf{w}_j(s)^\top \mathbf{z}_i)^2 \geq \frac{1}{\kappa} \sum_{j=1}^d \sum_{\mathbf{o}_i \in \mathcal{O}(s-1)} (\mathbf{w}_j(s)^\top \mathbf{o}_i)^2 \right\}.$$

We show in the next theorem that with high probability, $\mathcal{E}(s)$ is true for at least one “small” s , by showing that at every s where it is not true, the random removal procedure removes a corrupted point with probability at least $\kappa/(1 + \kappa)$.

Theorem 9 *With probability at least $1 - \gamma$, event $\mathcal{E}(s)$ is true for some $1 \leq s \leq s_0$, where*

$$s_0 \triangleq (1 + \epsilon) \frac{(1 + \kappa)\lambda n}{\kappa}; \quad \epsilon = \frac{16(1 + \kappa) \log(1/\gamma)}{\kappa \lambda n} + 4 \sqrt{\frac{(1 + \kappa) \log(1/\gamma)}{\kappa \lambda n}}.$$

Remark: When κ and λ are fixed, we have $s_0/n \rightarrow (1 + \kappa)\lambda/\kappa$. Therefore, $s_0 \leq t$ for $(1 + \kappa)\lambda < \kappa(1 - \lambda)$ and n large.

When $s_0 \geq n$, Theorem 9 holds trivially. Hence we focus on the case where $s_0 < n$. En route to proving this theorem, we first prove that when $\mathcal{E}(s)$ is not true, our procedure removes a corrupted point with high probability. To this end, let \mathcal{F}_s be the filtration generated by the set of events until stage s . Observe that $\mathcal{O}(s), \mathcal{Z}(s), \mathcal{Y}(s) \in \mathcal{F}_s$. Furthermore, since given $\mathcal{Y}(s)$, performing a PCA is deterministic, $\mathcal{E}(s + 1) \in \mathcal{F}_s$.

Theorem 10 *If the complimentary of $\mathcal{E}(s)$, denoted by $\mathcal{E}^c(s)$, is true, then*

$$\Pr(\{\bar{\tau}(s) \in \mathcal{O}(s - 1)\} | \mathcal{F}_{s-1}) > \frac{\kappa}{1 + \kappa}.$$

Proof: If $\mathcal{E}^c(s)$ is true, then

$$\sum_{j=1}^d \sum_{\mathbf{z}_i \in \mathcal{Z}(s-1)} (\mathbf{w}_j(s)^\top \mathbf{z}_i)^2 < \frac{1}{\kappa} \sum_{j=1}^d \sum_{\mathbf{o}_i \in \mathcal{O}(s-1)} (\mathbf{w}_j(s)^\top \mathbf{o}_i)^2,$$

which is equivalent to

$$\frac{\kappa}{1 + \kappa} \left[\sum_{\mathbf{z}_i \in \mathcal{Z}(s-1)} \sum_{j=1}^d (\mathbf{w}_j(s)^\top \mathbf{z}_i)^2 + \sum_{\mathbf{o}_i \in \mathcal{O}(s-1)} \sum_{j=1}^d (\mathbf{w}_j(s)^\top \mathbf{o}_i)^2 \right] < \sum_{\mathbf{o}_i \in \mathcal{O}(s-1)} \sum_{j=1}^d (\mathbf{w}_j(s)^\top \mathbf{o}_i)^2.$$

Note that

$$\begin{aligned} & \Pr(\{\bar{\tau}(s) \in \mathcal{O}(s - 1)\} | \mathcal{F}_{s-1}) \\ &= \sum_{\mathbf{o}_i \in \mathcal{O}(s-1)} \Pr(\bar{\tau}(s) = \mathbf{o}_i | \mathcal{F}_{s-1}) \\ &= \sum_{\mathbf{o}_i \in \mathcal{O}(s-1)} \frac{\sum_{j=1}^d (\mathbf{w}_j(s)^\top \mathbf{o}_i)^2}{\sum_{\mathbf{z}_i \in \mathcal{Z}(s-1)} \sum_{j=1}^d (\mathbf{w}_j(s)^\top \mathbf{z}_i)^2 + \sum_{\mathbf{o}_i \in \mathcal{O}(s-1)} \sum_{j=1}^d (\mathbf{w}_j(s)^\top \mathbf{o}_i)^2} \\ &> \frac{\kappa}{1 + \kappa}. \end{aligned}$$

Here, the second equality follows from the definition of the algorithm, and in particular, that in stage s , we remove a point \mathbf{y} with probability proportional to $\sum_{j=1}^d (\mathbf{w}_j(s)^\top \mathbf{y})^2$, and independent of other events. \blacksquare

As a consequence of this theorem, we can now prove Theorem 9. The intuition is rather straightforward: if the events were independent from one step to the next, then since “expected corrupted points removed” is at least $\kappa/(1 + \kappa)$, then after $s_0 = (1 + \epsilon)(1 + \kappa)\lambda n/\kappa$ steps, with exponentially high probability all the outliers would be removed, and hence we would have a good event with high probability, for some $s \leq s_0$. Since subsequent steps are not independent, we have to rely on martingale arguments.

Let $T = \min\{s | \mathcal{E}(s) \text{ is true}\}$. Note that since $\mathcal{E}(s) \in \mathcal{F}_{s-1}$, we have $\{T > s\} \in \mathcal{F}_{s-1}$. Define the following random variable

$$X_s = \begin{cases} |\mathcal{O}(T - 1)| + \frac{\kappa(T-1)}{1+\kappa}, & \text{if } T \leq s; \\ |\mathcal{O}(s)| + \frac{\kappa s}{1+\kappa}, & \text{if } T > s. \end{cases}$$

Lemma 11 $\{X_s, \mathcal{F}_s\}$ is a supermartingale.

Proof: The proof essentially follows from the definition of X_s , and the fact that if $\mathcal{E}(s)$ is true, then $|\mathcal{O}(s)|$ decreases by one with probability $\kappa/(1 + \kappa)$. The full details are deferred to the full version (Xu et al., 2010). \blacksquare

From here, the proof of Theorem 9 follows straightforwardly.

Proof: Note that

$$\Pr\left(\bigcap_{s=1}^{s_0} \mathcal{E}(s)^c\right) = \Pr(T > s_0) \leq \Pr\left(X_{s_0} \geq \frac{\kappa s_0}{1 + \kappa}\right) = \Pr(X_{s_0} \geq (1 + \epsilon)\lambda n), \quad (3)$$

where the inequality is due to $|\mathcal{O}(s)|$ being non-negative. Recall that $X_0 = \lambda n$. Thus the probability that no good events occur before step s_0 is at most the probability that a supermartingale with bounded increments increases in value by a constant factor of $(1 + \epsilon)$, from λn to $(1 + \epsilon)\lambda n$. An appeal to Azuma’s inequality shows that this is exponentially unlikely. The details are left to the long version (Xu et al., 2010). \blacksquare

4.3 Step 3

Let $\bar{\mathbf{w}}_1, \dots, \bar{\mathbf{w}}_d$ be the eigenvectors corresponding to the d largest eigenvalues of AA^\top , i.e., the optimal solution. Let $\mathbf{w}_1^*, \dots, \mathbf{w}_d^*$ be the output of the algorithm. Let $\mathbf{w}_1(s), \dots, \mathbf{w}_d(s)$ be the candidate solution at stage s . Recall that $H(\mathbf{w}_1, \dots, \mathbf{w}_d) \triangleq \sum_{j=1}^d \|\mathbf{w}_j^\top A\|^2$, and for notational simplification, let $\bar{H} \triangleq H(\bar{\mathbf{w}}_1, \dots, \bar{\mathbf{w}}_d)$, $H_s \triangleq H(\mathbf{w}_1(s), \dots, \mathbf{w}_d(s))$, and $H^* \triangleq H(\mathbf{w}_1^*, \dots, \mathbf{w}_d^*)$.

The statement of the finite-sample and asymptotic theorems (Theorems 1 and 2, respectively) lower bound the expressed variance, E.V., which is the ratio H^*/\bar{H} . The final part of the proof accomplishes this in two main steps. First, Lemma 12 lower bounds H_s in terms of \bar{H} , where s is some step for which $\mathcal{E}(s)$ is true, i.e., the principal components found by the s^{th} step of the algorithm are “good.” By Theorem 9, we know that there is a “small” such s , with high probability. The final output of the algorithm, however, is only guaranteed to have a high value of the robust variance estimator, \bar{V} — that is, even if there is a “good” solution at some intermediate step s , we do not necessarily have a way of identifying it. Thus, the next step, Lemma 13, lower bounds the value of H^* in terms of the value H of *any* output $\mathbf{w}_1^*, \dots, \mathbf{w}_d^*$ that has a smaller value of the robust variance estimator.

We give the statement of all the intermediate results, leaving the details to the full version (Xu et al., 2010).

Lemma 12 *If $\mathcal{E}(s)$ is true for some $s \leq s_0$, and there exists $\epsilon_1, \epsilon_2, \bar{c}$ such that (I) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^{t-s_0} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V}\left(\frac{t-s_0}{t}\right) \right| \leq \epsilon_1$; (II) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{x}_i|^2 - 1 \right| \leq \epsilon_2$; (III) $\sup_{\mathbf{w} \in \mathcal{S}_m} \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{n}_i|^2 \leq \bar{c}$, then*

$$\frac{1}{1+\kappa} \left[(1-\epsilon_1) \mathcal{V}\left(\frac{t-s_0}{t}\right) \bar{H} - 2\sqrt{(1+\epsilon_2)\bar{c}d\bar{H}} \right] \leq (1+\epsilon_2)H_s + 2\sqrt{(1+\epsilon_2)\bar{c}dH_s} + \bar{c}.$$

Lemma 13 *Fix a $\hat{t} \leq t$. If $\sum_{j=1}^d \bar{V}_{\hat{t}}(\mathbf{w}_j) \geq \sum_{j=1}^d \bar{V}_{\hat{t}}(\mathbf{w}'_j)$, and there exists $\epsilon_1, \epsilon_2, \bar{c}$ such that (I) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^{\hat{t}} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V}\left(\frac{\hat{t}}{t}\right) \right| \leq \epsilon_1$; (II) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^{\hat{t}-\frac{\lambda \hat{t}}{1-\lambda}} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V}\left(\frac{\hat{t}}{t} - \frac{\lambda}{1-\lambda}\right) \right| \leq \epsilon_1$; (III) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{x}_i|^2 - 1 \right| \leq \epsilon_2$; (IV) $\sup_{\mathbf{w} \in \mathcal{S}_m} \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{n}_i|^2 \leq \bar{c}$, then*

$$\begin{aligned} & (1-\epsilon_1) \mathcal{V}\left(\frac{\hat{t}}{t} - \frac{\lambda}{1-\lambda}\right) H(\mathbf{w}'_1, \dots, \mathbf{w}'_d) - 2\sqrt{(1+\epsilon_2)\bar{c}dH(\mathbf{w}'_1, \dots, \mathbf{w}'_d)} \\ & \leq (1+\epsilon_1)H(\mathbf{w}_1, \dots, \mathbf{w}_d) \mathcal{V}\left(\frac{\hat{t}}{t}\right) + 2\sqrt{(1+\epsilon_2)\bar{c}dH(\mathbf{w}_1, \dots, \mathbf{w}_d)} + \bar{c}. \end{aligned}$$

Theorem 14 *If $\bigcup_{s=1}^{s_0} \mathcal{E}(s)$ is true, and there exists $\epsilon_1 < 1, \epsilon_2, \bar{c}$ such that (I) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^{t-s_0} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V}\left(\frac{t-s_0}{t}\right) \right| \leq \epsilon_1$; (II) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^{\hat{t}} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V}\left(\frac{\hat{t}}{t}\right) \right| \leq \epsilon_1$; (III) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^{\hat{t}-\frac{\lambda \hat{t}}{1-\lambda}} |\mathbf{w}^\top \mathbf{x}_{(i)}|^2 - \mathcal{V}\left(\frac{\hat{t}}{t} - \frac{\lambda}{1-\lambda}\right) \right| \leq \epsilon_1$; (IV) $\sup_{\mathbf{w} \in \mathcal{S}_d} \left| \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{x}_i|^2 - 1 \right| \leq \epsilon_2$; (V) $\sup_{\mathbf{w} \in \mathcal{S}_m} \frac{1}{t} \sum_{i=1}^t |\mathbf{w}^\top \mathbf{n}_i|^2 \leq \bar{c}$, then*

$$\begin{aligned} \frac{H^*}{\bar{H}} & \geq \frac{(1-\epsilon_1)^2 \mathcal{V}\left(\frac{\hat{t}}{t} - \frac{\lambda}{1-\lambda}\right) \mathcal{V}\left(\frac{t-s_0}{t}\right)}{(1+\epsilon_1)(1+\epsilon_2)(1+\kappa) \mathcal{V}\left(\frac{\hat{t}}{t}\right)} \\ & - \left[\frac{(2\kappa+4)(1-\epsilon_1) \mathcal{V}\left(\frac{\hat{t}}{t} - \frac{\lambda}{1-\lambda}\right) \sqrt{(1+\epsilon_2)\bar{c}d} + 4(1+\kappa)(1+\epsilon_2) \sqrt{(1+\epsilon_2)\bar{c}d}}{(1+\epsilon_1)(1+\epsilon_2)(1+\kappa) \mathcal{V}\left(\frac{\hat{t}}{t}\right)} \right] (\bar{H})^{-1/2} \quad (4) \\ & - \left[\frac{(1-\epsilon_1) \mathcal{V}\left(\frac{\hat{t}}{t} - \frac{\lambda}{1-\lambda}\right) \bar{c} + (1+\epsilon_2)\bar{c}}{(1+\epsilon_1)(1+\epsilon_2) \mathcal{V}\left(\frac{\hat{t}}{t}\right)} \right] (\bar{H})^{-1}. \end{aligned}$$

By bounding all diminishing terms in the r.h.s. of (4), it reduces to Theorem 1. Theorem 2 follows immediately.

5 Concluding Remarks

In this paper we investigated the dimensionality-reduction problem in the case where the number and the dimensionality of samples are of the same magnitude, and a constant fraction of the points are

arbitrarily corrupted (perhaps maliciously so). We proposed a High-dimensional Robust Principal Component Analysis algorithm that is tractable, robust to corrupted points, easily kernelizable and asymptotically optimal. The algorithm iteratively finds a set of PCs using standard PCA and subsequently removes a point randomly with a probability proportional to its expressed variance. We provided both theoretical guarantees and favorable simulation results about the performance of the proposed algorithm.

To the best of our knowledge, previous efforts to extend existing robust PCA algorithms to the high-dimensional case were unsuccessful. Such algorithms are designed for low dimensional data sets where the observations significantly outnumber the variables of each dimension. When applied to high-dimensional data sets, they either lose statistical consistency due to lack of sufficient observations, or become intractable. This motivates our work of proposing a new robust PCA algorithm that takes into account the inherent difficulty in analyzing high-dimensional data.

Acknowledgement

We thank the anonymous reviewers for their insightful comments. Constantine Caramanis was partially supported by NSF grants EFRI-0735905, CNS-0721532, CNS-0831580, and DTRA grant HDTRA1-08-0029. Shie Mannor was supported by a Horev Fellowship, by the European Union under a Marie-Curie Reintegration Grant and by the Israel Science Foundation (contract 890015).

References

- Barnett, V. (1976). The ordering of multivariate data. *Journal of Royal Statistics Society Series, A*, 138, 318–344.
- Barnett, V., & Lewis, T. (1978). *Outliers in statistical data*. Wiley, New York.
- Bebbington, A. (1978). A method of bivariate trimming for robust estimation of the correlation coefficient. *Applied Statistics*, 27, 221–228.
- Brubaker, S. C. (2009). Robust PCA and clustering on noisy mixtures. *Proceedings of the Nineteenth Annual ACM -SIAM Symposium on Discrete Algorithms* (pp. 1078–1087).
- Candès, E., Li, X., Ma, Y., & Wright, J. (2009). Robust principal component analysis? ArXiv:0912.3599.
- Candès, E., & Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9, 717–772.
- Chandrasekaran, V., Sanghavi, S., Parrilo, P., & Willsky, A. (2009). Rank-sparsity incoherence for matrix decomposition. ArXiv:0906.2220.
- Croux, C., Filzmoser, P., & Oliveira, M. (2007). Algorithms for Projection–Pursuit robust principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 87, 218–225.
- Croux, C., & Hasebroeck, G. (2000). Principal component analysis based on robust estimators of the covariance or correlation matrix: Influence functions and efficiencies. *Biometrika*, 87, 603–618.
- Croux, C., & Ruiz-Gazen, A. (2005). High breakdown estimators for principal components: the projection-pursuit approach revisited. *Journal of Multivariate Analysis*, 95, 206–226.
- David, H. (1981). *Order statistics*. Wiley, New York.
- Davidson, K., & Szarek, S. (2001). Local operator theory, random matrices and banach spaces. *Handbook on the Geometry of Banach Spaces* (pp. 317–366). Elsevier.
- De la Torre, F., & Black, M. J. (2001). Robust principal component analysis for computer vision. *Proceedings of the Eighth International Conference on Computer Vision (ICCV'01)* (pp. 362–369).
- De la Torre, F., & Black, M. J. (2003). A framework for robust subspace learning. *International Journal of Computer Vision*, 54, 117–142.
- Dempster, A., & Gasko-Green, M. (1981). New tools for residual analysis. *The Annals of Statistics*, 9, 945–959.

- Devlin, S. J., Gnanadesikan, R., & Kettenring, J. R. (1975). Robust estimation and outlier detection with correlation coefficients. *Biometrika*, *62*, 531–545.
- Devlin, S. J., Gnanadesikan, R., & Kettenring, J. R. (1981). Robust estimation of dispersion matrices and principal components. *Journal of the American Statistical Association*, *76*, 354–362.
- Donoho, D. L. (1982). Breakdown properties of multivariate location estimators. Qualifying paper, Harvard University.
- Donoho, D. L. (2000). High-dimensional data analysis: The curses and blessings of dimensionality. American Math. Society Lecture—Math. Challenges of the 21st Century.
- Gnanadesikan, R., & Kettenring, J. R. (1972). Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics*, *28*, 81–124.
- Helbling, J. (1983). *Ellipsoïdes minimaux de couverture en statistique multivariée*. Doctoral dissertation, Ecole Polytechnique Fédérale de Lausanne, Switzerland.
- Huber, P. J. (1981). *Robust statistics*. John Wiley & Sons, New York.
- Hubert, M., Rousseeuw, P. J., & Branden, K. (2005). ROBPCA: A new approach to robust principal component analysis. *Technometrics*, *47*, 64–79.
- Johnstone, I. M. (2001). On the distribution of the largest eigenvalue in principal components analysis. *The Annals of Statistics*, *29*, 295–327.
- Jolliffe, I. T. (1986). *Principal component analysis*. Springer Series in Statistics, Berlin: Springer.
- Li, G., & Chen, Z. (1985). Projection-pursuit approach to robust dispersion matrices and principal components: Primary theory and monte carlo. *Journal of the American Statistical Association*, *80*, 759–766.
- Maronna, R. (1976). Robust M-estimators of multivariate location and scatter. *The Annals of Statistics*, *4*, 51–67.
- Mendelson, S., & Pajor, A. (2006). On singular values of matrices with independent rows. *Bernoulli*, *12*, 761–773.
- Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, *2*, 559–572.
- Recht, B. (2009). A simpler approach to matrix completion. ArXiv: 0910.0651.
- Recht, B., Fazel, M., & Parrilo, P. (2010). Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization. To appear in *SIAM Review*.
- Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, *79*, 871–880.
- Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. MIT Press.
- Schölkopf, B., Smola, A. J., & Müller, K. R. (1999). Kernel principal component analysis. *Advances in kernel Methods – Support Vector Learning* (pp. 327–352). MIT Press, Cambridge, MA.
- Titterton, D. (1978). Estimation of correlation coefficients by ellipsoidal trimming. *Applied Statistics*, *27*, 227–234.
- Xu, H., Caramanis, C., & Mannor, S. (2010). Principal component analysis with contaminated data: The high dimensional case. ArXiv: 1002.4658.
- Xu, L., & Yuille, A. L. (1995). Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Transactions on Neural Networks*, *6*, 131–143.
- Yang, T. N., & Wang, S. D. (1999). Robust algorithms for principal component analysis. *Pattern Recognition Letters*, *20*, 927–933.

Robustness and Generalization

Huan Xu

The University of Texas at Austin
huan.xu@mail.utexas.edu

Shie Mannor

Technion, Israel Institute of Technology
shie@ee.technion.ac.il

Abstract

We derive generalization bounds for learning algorithms based on their robustness: the property that if a testing sample is “similar” to a training sample, then the testing error is close to the training error. This provides a novel approach, different from the complexity or stability arguments, to study generalization of learning algorithms. We further show that a weak notion of robustness is both sufficient and necessary for generalizability, which implies that robustness is a fundamental property for learning algorithms to work.

1 Introduction

The key issue in the task of learning from a set of observed samples is the estimation of the *risk* (i.e., generalization error) of learning algorithms. Typically, since the learned hypothesis *depends on* the training data, its empirical measurement (i.e., training error) provides an optimistically biased estimation, especially when the number of training samples is small. Several approaches have been proposed to bound the deviation of the risk from its empirical measurement, among which methods based on uniform convergence and stability are most widely used.

Uniform convergence of empirical quantities to their mean (e.g., Vapnik and Chervonenkis 1974; 1991) provides ways to bound the gap between the expected risk and the empirical risk by the complexity of the hypothesis set. Examples to complexity measures are the Vapnik-Chervonenkis (VC) dimension (e.g., Vapnik & Chervonenkis, 1991; Evgeniou et al., 2000), the fat-shattering dimension (e.g., Alon et al., 1997; Bartlett, 1998), and the Rademacher complexity (Bartlett & Mendelson, 2002; Bartlett et al., 2005). Another well-known approach is based on *stability*. An algorithm is stable if its output remains “similar” for different sets of training samples that are identical up to removal or change of a single sample. The first results that relate stability to generalizability track back to Devroye and Wagner (1979a; 1979b). Later, McDiarmid’s concentration inequalities (McDiarmid, 1989) facilitated new bounds on generalization error (e.g., Bousquet & Elisseeff, 2002; Poggio et al., 2004; Mukherjee et al., 2006).

In this paper we explore a different approach which we term *algorithmic robustness*. Briefly speaking, an algorithm is robust if its solution has the following property: it achieves “similar” performance on a testing sample and a training sample that are “close”. This notion of robustness is rooted in *robust optimization* (Ben-Tal & Nemirovski, 1998; Ben-Tal & Nemirovski, 1999; Bertsimas & Sim, 2004) where a decision maker aims to find a solution x that minimizes a (parameterized) cost function $f(x, \xi)$ with the knowledge that the unknown true parameter ξ may deviate from the observed parameter $\hat{\xi}$. Hence, instead of solving $\min_x f(x, \hat{\xi})$ one solves $\min_x [\max_{\xi \in \Delta} f(x, \xi)]$, where Δ includes all possible realizations of ξ . Robust optimization was introduced in machine learning tasks to handle exogenous noise (e.g., Bhattacharyya et al., 2004; Shivaswamy et al., 2006; Globerson & Roweis, 2006), i.e., the learning algorithm only has access to inaccurate observation of training samples. Later on, Xu et al. (2010; 2009) showed that both Support Vector Machine(SVM) and Lasso have robust optimization interpretation, i.e., they can be reformulated as

$$\min_{h \in \mathcal{H}} \max_{(\delta_1, \dots, \delta_n) \in \Delta} \sum_{i=1}^n l(h, z_i + \delta_i),$$

for some Δ . Here z_i are the observed training samples and $l(\cdot, \cdot)$ is the loss function (hinge-loss for SVM, and squared loss for Lasso), which means that SVM and Lasso essentially minimize the

empirical error under the worst possible perturbation. Indeed, as Xu et al. (2010; 2009) showed, this reformulation leads to requiring that the loss of a sample “close” to z_i is small, which further implies statistical consistency of these two algorithms. In this paper we adopt this approach and study the (finite sample) generalization ability of learning algorithms by investigating the loss of learned hypotheses on samples that slightly deviate from training samples.

Of special interest is that robustness is more than just another way to establish generalization bounds. Indeed, we show that a weaker notion of robustness is a *necessary and sufficient* condition of (asymptotic) generalizability of (general) learning algorithms. While it is known having a finite VC-dimension (Vapnik & Chervonenkis, 1991) or equivalently being $\text{CVEEEE}_{l_{oo}}$ stable (Mukherjee et al., 2006) is necessary and sufficient for the Empirical Risk Minimization (ERM) to generalize, much less is known in the general case. Recently, Shalev-Shwartz et al. (2009) proposed a weaker notion of stability that is necessary and sufficient for a learning algorithm to be consistent and generalizing, provided that the problem itself is *learnable*. However, learnability requires that the *convergence rate is uniform* with respect to all distributions, and is hence a fairly strong assumption. In particular, the standard supervised learning setup where the hypothesis set is the set of measurable functions is *not* learnable since no algorithm can achieve a uniform convergence rate (cf. Devroye et al., 1996). Indeed, as Shalev-Shwartz et al. (2009) stated, for supervised learning problems learnability is equivalent to the generalizability of ERM, and hence reduce to the aforementioned results on ERM algorithms.

In particular, our main contributions are the following:

1. We propose a notion of algorithmic robustness. Algorithmic robustness is a desired property for a learning algorithm since it implies a lack of sensitivity to (small) disturbances in the training data.
2. Based on the notion of algorithmic robustness, we derive generalization bounds for IID samples.
3. To illustrate the applicability of the notion of algorithmic robustness, we provide some examples of robust algorithms, including SVM, Lasso, feed-forward neural networks and PCA.
4. We propose a weaker notion of robustness and show that it is both necessary and sufficient for a learning algorithm to generalize. This implies that robustness is an essential property needed for a learning algorithm to work.

Note that while stability and robustness are similar on an intuitive level, there is a difference between the two: stability requires that identical training sets with a single sample removed lead to similar prediction rules, whereas robustness requires that a prediction rule has comparable performance if tested on a sample close to a training sample. Simply put, stability compares two prediction rules, whereas robustness investigates one prediction rule.

This paper is organized as follows. We define the notion of robustness in Section 2, and prove generalization bounds for robust algorithms in Section 3. In Section 4 we propose a relaxed notion of robustness, which is termed as pseudo-robustness, and provide corresponding generalization bounds. Examples of learning algorithms that are robust or pseudo-robust are provided in Section 5. Finally, we show that robustness is necessary and sufficient for generalizability in Section 6. Due to space constraints, some of the proofs are deferred to the full version (Xu & Mannor, 2010).

1.1 Preliminaries

We consider the following general learning model: a set of training samples are given, and the goal is to pick a hypothesis from a hypothesis set. Unless otherwise mentioned, throughout this paper the size of training set is fixed as n . Therefore, we drop the dependence of parameters on the number of training samples, while it should be understood that parameters may vary with the number of training samples. We use \mathcal{Z} and \mathcal{H} to denote the set from which each sample is drawn, and the hypothesis set, respectively. Throughout the paper we use \mathbf{s} to denote the training sample set consists of n training samples (s_1, \dots, s_n) . A learning algorithm \mathcal{A} is thus a mapping from \mathcal{Z}^n to \mathcal{H} . We use $\mathcal{A}_{\mathbf{s}}$ to represent the hypothesis learned (given training set \mathbf{s}). For each hypothesis $h \in \mathcal{H}$ and a point $z \in \mathcal{Z}$, there is an associated loss $l(h, z)$. We ignore the issue of measurability and further assume that $l(h, z)$ is non-negative and upper-bounded uniformly by a scalar M .

In the special case of supervised learning, the sample space can be decomposed as $\mathcal{Z} = \mathcal{Y} \times \mathcal{X}$, and the goal is to learn a mapping from \mathcal{X} to \mathcal{Y} , i.e., to predict the y-component given x-component. We hence use $\mathcal{A}_{\mathbf{s}}(x)$ to represent the prediction of $x \in \mathcal{X}$ if trained on \mathbf{s} . We call \mathcal{X} the input space and \mathcal{Y} the output space. The output space can either be $\mathcal{Y} = \{-1, +1\}$ for a classification problem, or $\mathcal{Y} = \mathbb{R}$ for a regression problem. We use $|_x$ and $|_y$ to denote the x -component and y -component

of a point. For example, $s_{i|x}$ is the x -component of s_i . To simplify notations, for a scalar c , we use $[c]^+$ to represent its non-negative part, i.e., $[c]^+ \triangleq \max(0, c)$.

We recall the following standard notion of covering number from van der Vaart and Wellner (2000).

Definition 1 (cf. van der Vaart & Wellner, 2000) For a metric space S, ρ and $T \subset S$ we say that $\hat{T} \subset S$ is an ϵ -cover of T , if $\forall t \in T, \exists \hat{t} \in \hat{T}$ such that $\rho(t, \hat{t}) \leq \epsilon$. The ϵ -covering number of T is

$$\mathcal{N}(\epsilon, T, \rho) = \min\{|\hat{T}| : \hat{T} \text{ is an } \epsilon\text{-cover of } T\}.$$

2 Robustness of Learning Algorithms

Before providing a precise definition of what we mean by “robustness” of an algorithm, we provide some motivating examples which share a common property: if a testing sample is close to a training sample, then the testing error is also close, a property we will later formalize as “robustness”.

We first consider large-margin classifiers: Let the loss function be $l(\mathcal{A}_s, z) = \mathbf{1}(\mathcal{A}_s(z|x) \neq z|y)$. Fix $\gamma > 0$. An algorithm \mathcal{A}_s has a margin γ if for $j = 1, \dots, n$

$$\mathcal{A}_s(x) = \mathcal{A}_s(s_{j|x}); \quad \forall x : \|x - s_{j|x}\|_2 < \gamma.$$

That is, any training sample is at least γ away from the classification boundary.

Example 1 Fix $\gamma > 0$ and put $K = 2\mathcal{N}(\gamma/2, \mathcal{X}, \|\cdot\|_2)$. If \mathcal{A}_s has a margin γ , then \mathcal{Z} can be partitioned into K disjoint sets, denoted by $\{C_i\}_{i=1}^K$, such that if s_j and $z \in \mathcal{Z}$ belong to a same C_i , then $|l(\mathcal{A}_s, s_j) - l(\mathcal{A}_s, z)| = 0$.

Proof: By the definition of covering number, we can partition \mathcal{X} into $\mathcal{N}(\gamma/2, \mathcal{X}, \|\cdot\|_2)$ subsets (denoted \hat{X}_i) such that each subset has a diameter less or equal to γ . Further, \mathcal{Y} can be partitioned to $\{-1\}$ and $\{+1\}$. Thus, we can partition \mathcal{Z} into $2\mathcal{N}(\gamma/2, \mathcal{X}, \|\cdot\|_2)$ subsets such that if z_1, z_2 belong to a same subset, then $y_{1|y} = y_{2|y}$ and $\|x_{1|y} - x_{2|y}\| \leq \gamma$. By the definition of the margin, this guarantees that if s_j and $z \in \mathcal{Z}$ belong to a same C_i , then $|l(\mathcal{A}_s, s_j) - l(\mathcal{A}_s, z)| = 0$. ■

The next example is a linear regression algorithm. Let the loss function be $l(\mathcal{A}_s, z) = |z|_y - \mathcal{A}_s(z|x)|$, and let \mathcal{X} be a bounded subset of \mathbb{R}^m and fix $c > 0$. The norm-constrained linear regression algorithm is

$$\mathcal{A}_s = \min_{w \in \mathbb{R}^m : \|w\|_2 \leq c} \sum_{i=1}^n |s_{i|y} - w^\top s_{i|x}|, \quad (1)$$

i.e., minimizing the empirical error among all linear classifiers whose norm is bounded.

Example 2 Fix $\epsilon > 0$ and let $K = \mathcal{N}(\epsilon/2, \mathcal{X}, \|\cdot\|_2) \times \mathcal{N}(\epsilon/2, \mathcal{Y}, |\cdot|)$. Consider the algorithm as in (1). The set \mathcal{Z} can be partitioned into K disjoint sets, such that if s_j and $z \in \mathcal{Z}$ belong to a same C_i , then

$$|l(\mathcal{A}_s, s_j) - l(\mathcal{A}_s, z)| \leq (c + 1)\epsilon.$$

Proof: Similarly to the previous example, we can partition \mathcal{Z} to $\mathcal{N}(\epsilon/2, \mathcal{X}, \|\cdot\|_2) \times \mathcal{N}(\epsilon/2, \mathcal{Y}, |\cdot|)$ subsets, such that if z_1, z_2 belong to a same C_i , then $\|z_{1|x} - z_{2|x}\|_2 \leq \epsilon$, and $|z_{1|y} - z_{2|y}| \leq \epsilon$. Since $\|w\|_2 \leq c$, we have

$$\begin{aligned} |l(w, z_1) - l(w, z_2)| &= \left| |z_{1|y} - w^\top z_{1|x}| - |z_{2|y} - w^\top z_{2|x}| \right| \\ &\leq \left| (z_{1|y} - w^\top z_{1|x}) - (z_{2|y} - w^\top z_{2|x}) \right| \\ &\leq |z_{1|y} - z_{2|y}| + \|w\|_2 \|z_{1|x} - z_{2|x}\|_2 \\ &\leq (1 + c)\epsilon, \end{aligned}$$

whenever z_1, z_2 belong to a same C_i . ■

The two motivating examples both share a property: we can partition the sample set into finite subsets, such that if a new sample falls into the same subset as a training sample, then the loss of the former is close to the loss of the latter. We call an algorithm having this property “robust.”

Definition 2 Algorithm \mathcal{A} is $(K, \epsilon(\mathbf{s}))$ robust if \mathcal{Z} can be partitioned into K disjoint sets, denoted by $\{C_i\}_{i=1}^K$, such that $\forall \mathbf{s} \in \mathbf{s}$,

$$s, z \in C_i, \implies |l(\mathcal{A}_s, s) - l(\mathcal{A}_s, z)| \leq \epsilon(\mathbf{s}). \quad (2)$$

In the definition, both K and the partition sets $\{C_i\}_{i=1}^K$ do not depend on the training set \mathbf{s} . Note that the definition of robustness requires that (2) holds for every training sample. Indeed, we can relax the definition, so that the condition needs only hold for a subset of training samples. We call an algorithm having this property “pseudo robust.” See Section 4 for details.

3 Generalization Properties of Robust Algorithms

In this section we investigate generalization of robust algorithms. In particular, in the following subsections we derive PAC bounds for robust algorithms under two different conditions: (1) The ubiquitous learning setup where the samples are i.i.d. and the goal of learning is to minimize expected loss. (2) The learning goal is to minimize quantile loss. Indeed, the fact that we can provide results in (2) indicates the fundamental nature of robustness as a property of learning algorithms.

3.1 IID samples and expected loss

In this section, we consider the standard learning setup, i.e., the sample set \mathbf{s} consists of n i.i.d. samples generated by an unknown distribution μ , and the goal of learning is to minimize expected test loss. Let $\hat{l}(\cdot)$ and $l_{\text{emp}}(\cdot)$ denote the expected error and the training error, i.e.,

$$\hat{l}(\mathcal{A}_{\mathbf{s}}) \triangleq \mathbb{E}_{z \sim \mu} l(\mathcal{A}_{\mathbf{s}}, z); \quad l_{\text{emp}}(\mathcal{A}_{\mathbf{s}}) \triangleq \frac{1}{n} \sum_{s_i \in \mathbf{s}} l(\mathcal{A}_{\mathbf{s}}, s_i).$$

Recall that the loss function $l(\cdot, \cdot)$ is upper bounded by M .

Theorem 3 *If \mathbf{s} consists of n i.i.d. samples, and \mathcal{A} is $(K, \epsilon(\mathbf{s}))$ -robust, then for any $\delta > 0$, with probability at least $1 - \delta$,*

$$\left| \hat{l}(\mathcal{A}_{\mathbf{s}}) - l_{\text{emp}}(\mathcal{A}_{\mathbf{s}}) \right| \leq \epsilon(\mathbf{s}) + M \sqrt{\frac{2K \ln 2 + 2 \ln(1/\delta)}{n}}.$$

Proof: Let N_i be the set of index of points of \mathbf{s} that fall into C_i . Note that $(|N_1|, \dots, |N_K|)$ is an IID multinomial random variable with parameters n and $(\mu(C_1), \dots, \mu(C_K))$. The following holds by the Bretaganolle-Huber-Carol inequality (cf Proposition A6.6 of (van der Vaart & Wellner, 2000)):

$$\Pr \left\{ \sum_{i=1}^K \left| \frac{|N_i|}{n} - \mu(C_i) \right| \geq \lambda \right\} \leq 2^K \exp\left(\frac{-n\lambda^2}{2}\right).$$

Hence, the following holds with probability at least $1 - \delta$,

$$\sum_{i=1}^K \left| \frac{|N_i|}{n} - \mu(C_i) \right| \leq \sqrt{\frac{2K \ln 2 + 2 \ln(1/\delta)}{n}}. \quad (3)$$

We have

$$\begin{aligned} & \left| \hat{l}(\mathcal{A}_{\mathbf{s}}) - l_{\text{emp}}(\mathcal{A}_{\mathbf{s}}) \right| \\ &= \left| \sum_{i=1}^K \mathbb{E}(l(\mathcal{A}_{\mathbf{s}}, z) | z \in C_i) \mu(C_i) - \frac{1}{n} \sum_{i=1}^n l(\mathcal{A}_{\mathbf{s}}, s_i) \right| \\ &\stackrel{(a)}{\leq} \left| \sum_{i=1}^K \mathbb{E}(l(\mathcal{A}_{\mathbf{s}}, z) | z \in C_i) \frac{|N_i|}{n} - \frac{1}{n} \sum_{i=1}^n l(\mathcal{A}_{\mathbf{s}}, s_i) \right| \\ &\quad + \left| \sum_{i=1}^K \mathbb{E}(l(\mathcal{A}_{\mathbf{s}}, z) | z \in C_i) \mu(C_i) - \sum_{i=1}^K \mathbb{E}(l(\mathcal{A}_{\mathbf{s}}, z) | z \in C_i) \frac{|N_i|}{n} \right| \\ &\stackrel{(b)}{\leq} \left| \frac{1}{n} \sum_{i=1}^K \sum_{j \in N_i} \max_{z_2 \in C_i} |l(\mathcal{A}_{\mathbf{s}}, s_j) - l(\mathcal{A}_{\mathbf{s}}, z_2)| \right| + \left| \max_{z \in \mathcal{Z}} |l(\mathcal{A}_{\mathbf{s}}, z)| \sum_{i=1}^K \left| \frac{|N_i|}{n} - \mu(C_i) \right| \right| \\ &\stackrel{(c)}{\leq} \epsilon(\mathbf{s}) + M \sum_{i=1}^K \left| \frac{|N_i|}{n} - \mu(C_i) \right|, \end{aligned} \quad (4)$$

where (a), (b), and (c) are due to the triangle inequality, the definition of N_i , and the definition of $\epsilon(\mathbf{s})$ and M , respectively. The right-hand-side of (4) is upper-bounded by $\epsilon(\mathbf{s}) + M \sqrt{\frac{2K \ln 2 + 2 \ln(1/\delta)}{n}}$ with probability at least $1 - \delta$ due to (3). The theorem follows. \blacksquare

Theorem 3 requires that we fix a K *a priori*. However, it is often worthwhile to consider adaptive K . For example, in the large-margin classification case, typically the margin is known only after \mathbf{s} is realized. That is, the value of K depends on \mathbf{s} . Because of this dependency, we need a generalization bound that holds uniformly for all K .

Corollary 4 *If \mathbf{s} consists of n i.i.d. samples, and \mathcal{A} is $(K, \epsilon_K(\mathbf{s}))$ robust for all $K \geq 1$, then for any $\delta > 0$, with probability at least $1 - \delta$,*

$$\left| \hat{l}(\mathcal{A}_{\mathbf{s}}) - l_{\text{emp}}(\mathcal{A}_{\mathbf{s}}) \right| \leq \inf_{K \geq 1} \left[\epsilon_K(\mathbf{s}) + M \sqrt{\frac{2K \ln 2 + 2 \ln \frac{K(K+1)}{\delta}}{n}} \right].$$

Proof: Let

$$E(K) \triangleq \left\{ \left| \hat{l}(\mathcal{A}_{\mathbf{s}}) - l_{\text{emp}}(\mathcal{A}_{\mathbf{s}}) \right| > \epsilon_K(\mathbf{s}) + M \sqrt{\frac{2K \ln 2 + 2 \ln \frac{K(K+1)}{\delta}}{n}} \right\}.$$

From Theorem 3 we have $\Pr(E(K)) \leq \delta/(K(K+1)) = \delta/K - \delta/(K+1)$. By the union bound we have

$$\Pr \left\{ \bigcup_{K \geq 1} E(K) \right\} \leq \sum_{K \geq 1} \Pr(E(K)) \leq \sum_{K \geq 1} \left[\frac{\delta}{K} - \frac{\delta}{K+1} \right] = \delta,$$

and the corollary follows. ■

If $\epsilon(\mathbf{s})$ does not depend on \mathbf{s} , we can sharpen the bound given in Corollary 4.

Corollary 5 *If \mathbf{s} consists of n i.i.d. samples, and \mathcal{A} is (K, ϵ_K) robust for all $K \geq 1$, then for any $\delta > 0$, with probability at least $1 - \delta$,*

$$\left| \hat{l}(\mathcal{A}_{\mathbf{s}}) - l_{\text{emp}}(\mathcal{A}_{\mathbf{s}}) \right| \leq \inf_{K \geq 1} \left[\epsilon_K + M \sqrt{\frac{2K \ln 2 + 2 \ln \frac{1}{\delta}}{n}} \right].$$

Proof: Take K^* that minimizes the right hand side, and note that it does not depend on \mathbf{s} . Therefore, plugging K^* into Theorem 3 establishes the corollary. ■

3.2 Quantile Loss

So far we considered the standard expected loss setup. In this section we consider some less extensively investigated loss functions, namely quantile value and truncated expectation (see the following for precise definitions). These loss functions are of interest because they are less sensitive to the presence of outliers than the standard average loss (Huber, 1981).

Definition 6 *For a non-negative random variable X , the β -quantile value is*

$$\mathbb{Q}^\beta(X) \triangleq \inf \{c \in \mathbb{R} : \Pr(X \leq c) \geq \beta\}.$$

The β -truncated mean is

$$\mathbb{T}^\beta(X) \triangleq \begin{cases} \mathbb{E}[X \cdot \mathbf{1}(X < \mathbb{Q}^\beta(X))] & \text{if } \Pr[X = \mathbb{Q}^\beta(X)] = 0; \\ \mathbb{E}[X \cdot \mathbf{1}(X < \mathbb{Q}^\beta(X))] + \frac{\beta - \Pr[X < \mathbb{Q}^\beta(X)]}{\Pr[X = \mathbb{Q}^\beta(X)]} \mathbb{Q}^\beta(X) & \text{otherwise.} \end{cases}$$

In words, the β -quantile loss is the smallest value that is larger or equal to X with probability at least β . The β -truncated mean is the contribution to the expectation of the leftmost β fraction of the distribution. For example, suppose X is supported on $\{c_1, \dots, c_{10}\}$ ($c_1 < c_2 < \dots < c_{10}$) and the probability of taking each value equals 0.1. Then the 0.63-quantile loss of X is c_7 , and the 0.63-truncated mean of X equals $0.1(\sum_{i=1}^6 c_i + 0.3c_7)$.

Given $h \in \mathcal{H}$, $\beta \in (0, 1)$, and a probability measure μ on \mathcal{Z} , let

$$\mathcal{Q}(h, \beta, \mu) \triangleq \mathbb{Q}^\beta(l(h, z)); \quad \text{where: } z \sim \mu;$$

and

$$\mathcal{T}(h, \beta, \mu) \triangleq \mathbb{T}^\beta(l(h, z)); \quad \text{where: } z \sim \mu;$$

i.e., the β -quantile value and β -truncated mean of the (random) testing error of hypothesis h if the testing sample follows distribution μ . We have the following theorem that is a special case of Theorem 10, hence we omit the proof.

Theorem 7 (Quantile Value & Truncated Mean) Suppose \mathbf{s} are n i.i.d. samples drawn according to μ , and denote the empirical distribution of \mathbf{s} by μ_{emp} . Let $\lambda_0 = \sqrt{\frac{2K \ln 2 + 2 \ln(1/\delta)}{n}}$. If $0 \leq \beta - \lambda_0 \leq \beta + \lambda_0 \leq 1$ and \mathcal{A} is $(K, \epsilon(\mathbf{s}))$ robust, then with probability at least $1 - \delta$, the followings hold

$$\begin{aligned} (I) \quad & \mathcal{Q}(\mathcal{A}_{\mathbf{s}}, \beta - \lambda_0, \mu_{\text{emp}}) - \epsilon(\mathbf{s}) \leq \mathcal{Q}(\mathcal{A}_{\mathbf{s}}, \beta, \mu) \leq \mathcal{Q}(\mathcal{A}_{\mathbf{s}}, \beta + \lambda_0, \mu_{\text{emp}}) + \epsilon(\mathbf{s}); \\ (II) \quad & \mathcal{T}(\mathcal{A}_{\mathbf{s}}, \beta - \lambda_0, \mu_{\text{emp}}) - \epsilon(\mathbf{s}) \leq \mathcal{T}(\mathcal{A}_{\mathbf{s}}, \beta, \mu) \leq \mathcal{T}(\mathcal{A}_{\mathbf{s}}, \beta + \lambda_0, \mu_{\text{emp}}) + \epsilon(\mathbf{s}). \end{aligned}$$

In words, Theorem 7 essentially means that with high probability, the β -quantile value/truncated mean of the testing error (recall that the testing error is a random variable) is (approximately) bounded by the $(\beta \pm \lambda_0)$ -quantile value/truncated mean of the empirical error, thus providing a way to estimate the quantile value/truncated expectation of the testing error based on empirical observations.

4 Pseudo Robustness

In this section we propose a relaxed definition of robustness that accounts for the case where Equation (2) holds for most of training samples, as opposed to Definition 2 where Equation (2) holds for all training samples. Recall that the size of training set is fixed as n .

Definition 8 Algorithm \mathcal{A} is $(K, \epsilon(\mathbf{s}), \hat{n}(\mathbf{s}))$ pseudo robust if \mathcal{Z} can be partitioned into K disjoint sets, denoted as $\{C_i\}_{i=1}^K$, and there exists a subset of training samples $\hat{\mathbf{s}}$ with $|\hat{\mathbf{s}}| = \hat{n}(\mathbf{s})$ such that $\forall s \in \hat{\mathbf{s}}$,

$$s, z \in C_i, \implies |l(\mathcal{A}_{\mathbf{s}}, s) - l(\mathcal{A}_{\mathbf{s}}, z)| \leq \epsilon(\mathbf{s}).$$

Observe that $(K, \epsilon(\mathbf{s}))$ -robust is equivalent to $(K, \epsilon(\mathbf{s}), n)$ pseudo robust.

Theorem 9 If \mathbf{s} consists of n i.i.d. samples, and \mathcal{A} is $(K, \epsilon(\mathbf{s}), \hat{n}(\mathbf{s}))$ pseudo robust, then for any $\delta > 0$, with probability at least $1 - \delta$,

$$\left| \hat{l}(\mathcal{A}_{\mathbf{s}}) - l_{\text{emp}}(\mathcal{A}_{\mathbf{s}}) \right| \leq \frac{\hat{n}(\mathbf{s})}{n} \epsilon(\mathbf{s}) + M \left(\frac{n - \hat{n}(\mathbf{s})}{n} + \sqrt{\frac{2K \ln 2 + 2 \ln(1/\delta)}{n}} \right).$$

Proof: Let N_i and \hat{N}_i be the set of indices of points of \mathbf{s} and $\hat{\mathbf{s}}$ that fall into the C_i , respectively. Similarly to the proof of Theorem 3, we note that $(|N_1|, \dots, |N_K|)$ is an IID multinomial random variable with parameters n and $(\mu(C_1), \dots, \mu(C_K))$. And hence due to Breteganolle-Huber-Carol Inequality, the following holds with probability at least $1 - \delta$,

$$\sum_{i=1}^K \left| \frac{|N_i|}{n} - \mu(C_i) \right| \leq \sqrt{\frac{2K \ln 2 + 2 \ln(1/\delta)}{n}}. \quad (5)$$

Furthermore, we have

$$\begin{aligned} & \left| \hat{l}(\mathcal{A}_{\mathbf{s}}) - l_{\text{emp}}(\mathcal{A}_{\mathbf{s}}) \right| \\ &= \left| \sum_{i=1}^K \mathbb{E}(l(\mathcal{A}_{\mathbf{s}}, z) | z \in C_i) \mu(C_i) - \frac{1}{n} \sum_{i=1}^n l(\mathcal{A}_{\mathbf{s}}, s_i) \right| \\ &\leq \left| \sum_{i=1}^K \mathbb{E}(l(\mathcal{A}_{\mathbf{s}}, z) | z \in C_i) \frac{|N_i|}{n} - \frac{1}{n} \sum_{i=1}^n l(\mathcal{A}_{\mathbf{s}}, s_i) \right| \\ &\quad + \left| \sum_{i=1}^K \mathbb{E}(l(\mathcal{A}_{\mathbf{s}}, z) | z \in C_i) \mu(C_i) - \sum_{i=1}^K \mathbb{E}(l(\mathcal{A}_{\mathbf{s}}, z) | z \in C_i) \frac{|N_i|}{n} \right| \\ &\leq \left| \frac{1}{n} \sum_{i=1}^K [|N_i| \times \mathbb{E}(l(\mathcal{A}_{\mathbf{s}}, z) | z \in C_i) - \sum_{j \in \hat{N}_i} l(\mathcal{A}_{\mathbf{s}}, s_j) - \sum_{j \in N_i, j \notin \hat{N}_i} l(\mathcal{A}_{\mathbf{s}}, s_j)] \right| \\ &\quad + \left| \max_{z \in \mathcal{Z}} |l(\mathcal{A}_{\mathbf{s}}, z)| \sum_{i=1}^K \left| \frac{|N_i|}{n} - \mu(C_i) \right| \right|. \end{aligned}$$

Note that due to the triangle inequality as well as the assumption that the loss is non-negative and upper bounded by M , the right-hand side can be upper bounded by

$$\begin{aligned} & \left| \frac{1}{n} \sum_{i=1}^K \sum_{j \in \hat{N}_i} \max_{z_2 \in C_i} |l(\mathcal{A}_s, s_j) - l(\mathcal{A}_s, z_2)| \right| + \left| \frac{1}{n} \sum_{i=1}^K \sum_{j \in N_i, j \notin \hat{N}_i} \max_{z_2 \in C_i} |l(\mathcal{A}_s, s_j) - l(\mathcal{A}_s, z_2)| \right| \\ & \quad + M \sum_{i=1}^K \left| \frac{|N_i|}{n} - \mu(C_i) \right| \\ & \leq \frac{\hat{n}(\mathbf{s})}{n} \epsilon(\mathbf{s}) + \frac{n - \hat{n}(\mathbf{s})}{n} M + M \sum_{i=1}^K \left| \frac{|N_i|}{n} - \mu(C_i) \right|. \end{aligned}$$

where the inequality holds due to definition of N_i and \hat{N}_i . The theorem follows by applying (5). \blacksquare

Similarly, Theorem 7 can be generalized to the pseudo robust case. See the full version (Xu & Mannor, 2010) for the proof.

Theorem 10 (Quantile Value & Truncated Expectation) *Suppose \mathbf{s} has n samples drawn i.i.d. according to μ , and denote the empirical distribution of \mathbf{s} as μ_{emp} . Let $\lambda_0 = \sqrt{\frac{2K \ln 2 + 2 \ln(1/\delta)}{n}}$. Suppose $0 \leq \beta - \lambda_0 - (n - \hat{n})/n \leq \beta + \lambda_0 + (n - \hat{n})/n \leq 1$ and \mathcal{A} is $(K, \epsilon(\mathbf{s}), \hat{n}(\mathbf{s}))$ pseudo robust. Then with probability at least $1 - \delta$, the followings hold*

$$\begin{aligned} (I) \quad & \mathcal{Q} \left(\mathcal{A}_s, \beta - \lambda_0 - \frac{n - \hat{n}(\mathbf{s})}{n}, \mu_{\text{emp}} \right) - \epsilon(\mathbf{s}) \leq \mathcal{Q}(\mathcal{A}_s, \beta, \mu) \leq \mathcal{Q} \left(\mathcal{A}_s, \beta + \lambda_0 + \frac{n - \hat{n}(\mathbf{s})}{n}, \mu_{\text{emp}} \right) + \epsilon(\mathbf{s}); \\ (II) \quad & \mathcal{T} \left(\mathcal{A}_s, \beta - \lambda_0 - \frac{n - \hat{n}(\mathbf{s})}{n}, \mu_{\text{emp}} \right) - \epsilon(\mathbf{s}) \leq \mathcal{T}(\mathcal{A}_s, \beta, \mu) \leq \mathcal{T} \left(\mathcal{A}_s, \beta + \lambda_0 + \frac{n - \hat{n}(\mathbf{s})}{n}, \mu_{\text{emp}} \right) + \epsilon(\mathbf{s}). \end{aligned}$$

5 Examples of Robust Algorithms

In this section we provide some examples of robust algorithms. The proofs of the examples can be found in the full version (Xu & Mannor, 2010). Our first example is Majority Voting (MV) classification (cf Section 6.3 of Devroye et al., 1996) that partitions the input space \mathcal{X} and labels each partition set according to a majority vote of the training samples belonging to it.

Example 3 (Majority Voting) *Let $\mathcal{Y} = \{-1, +1\}$. Partition \mathcal{X} to $\mathcal{C}_1, \dots, \mathcal{C}_K$, and use $\mathcal{C}(x)$ to denote the set to which x belongs. A new sample $x_a \in \mathcal{X}$ is labeled by*

$$\mathcal{A}_s(x_a) \triangleq \begin{cases} 1, & \text{if } \sum_{s_i \in \mathcal{C}(x_a)} \mathbf{1}(s_{i|y} = 1) \geq \sum_{s_i \in \mathcal{C}(x_a)} \mathbf{1}(s_{i|y} = -1); \\ -1, & \text{otherwise.} \end{cases}$$

If the loss function is $l(\mathcal{A}_s, z) = f(z_{|y}, \mathcal{A}_s(z_{|x}))$ for some function f , then MV is $(2K, 0)$ robust.

MV algorithm has a natural partition of the sample space that makes it robust. Another class of robust algorithms are those that have approximately the same testing loss for testing samples that are close (in the sense of geometric distance) to each other, since we can partition the sample space with norm balls. The next theorem states that an algorithm is robust if two samples being close implies that they have similar testing error.

Theorem 11 *Fix $\gamma > 0$ and metric ρ of \mathcal{Z} . Suppose \mathcal{A} satisfies*

$$|l(\mathcal{A}_s, z_1) - l(\mathcal{A}_s, z_2)| \leq \epsilon(\mathbf{s}), \quad \forall z_1, z_2 : z_1 \in \mathbf{s}, \rho(z_1, z_2) \leq \gamma,$$

and $\mathcal{N}(\gamma/2, \mathcal{Z}, \rho) < \infty$. Then \mathcal{A} is $(\mathcal{N}(\gamma/2, \mathcal{Z}, \rho), \epsilon(\mathbf{s}))$ -robust.

Proof: Let $\{c_1, \dots, c_{\mathcal{N}(\gamma/2, \mathcal{Z}, \rho)}\}$ be a $\gamma/2$ -cover of \mathcal{Z} . whose existence is guaranteed by the definition of covering number. Let $\hat{C}_i = \{z \in \mathcal{Z} | \rho(z, c_i) \leq \gamma/2\}$, and $C_i = \hat{C}_i \cap (\bigcup_{j=1}^{i-1} \hat{C}_j)^c$. Thus, $C_1, \dots, C_{\mathcal{N}(\gamma/2, \mathcal{Z}, \rho)}$ is a partition of \mathcal{Z} , and satisfies

$$z_1, z_2 \in C_i \implies \rho(z_1, z_2) \leq \rho(z_1, c_i) + \rho(z_2, c_i) \leq \gamma.$$

Therefore,

$$|l(\mathcal{A}_s, z_1) - l(\mathcal{A}_s, z_2)| \leq \epsilon(\mathbf{s}), \quad \forall z_1, z_2 : z_1 \in \mathbf{s}, \rho(z_1, z_2) \leq \gamma,$$

implies

$$z_1 \in \mathbf{s}, z_2 \in C_i \implies |l(\mathcal{A}_{\mathbf{s}}, z_1) - l(\mathcal{A}_{\mathbf{s}}, z_2)| \leq \epsilon(\mathbf{s}),$$

and the theorem follows. \blacksquare

Theorem 11 immediately leads to the next example: if the testing error given the output of an algorithm is Lipschitz continuous, then the algorithm is robust.

Example 4 (Lipschitz continuous functions) *If \mathcal{Z} is compact w.r.t. metric ρ , $l(\mathcal{A}_{\mathbf{s}}, \cdot)$ is Lipschitz continuous with Lipschitz constant $c(\mathbf{s})$, i.e.,*

$$|l(\mathcal{A}_{\mathbf{s}}, z_1) - l(\mathcal{A}_{\mathbf{s}}, z_2)| \leq c(\mathbf{s})\rho(z_1, z_2), \quad \forall z_1, z_2 \in \mathcal{Z},$$

then \mathcal{A} is $(\mathcal{N}(\gamma/2, \mathcal{Z}, \rho), c(\mathbf{s})\gamma)$ -robust for all $\gamma > 0$.

Theorem 11 also implies that SVM, Lasso, feed-forward neural network and PCA are robust, as stated in Example 5 to Example 8. The proofs are deferred to Appendix.

Example 5 (Support Vector Machines) *Let \mathcal{X} be compact. Consider the standard SVM formulation (Cortes & Vapnik, 1995; Schölkopf & Smola, 2002)*

$$\begin{aligned} \text{Minimize:}_{\mathbf{w}, d} \quad & c\|w\|_{\mathcal{H}}^2 + \frac{1}{n} \sum_{i=1}^n \xi_i \\ \text{s. t.} \quad & 1 - s_{i|y}[\langle w, \phi(s_{i|x}) \rangle + d] \leq \xi_i; \\ & \xi_i \geq 0. \end{aligned}$$

Here $\phi(\cdot)$ is a feature mapping, $\|\cdot\|_{\mathcal{H}}$ is its RKHS kernel, and $k(\cdot, \cdot)$ is the kernel function. Let $l(\cdot, \cdot)$ be the hinge-loss, i.e., $l((w, d), z) = [1 - z_{|y}(\langle w, \phi(z_{|x}) \rangle + d)]^+$, and define $f_{\mathcal{H}}(\gamma) \triangleq \max_{\mathbf{a}, \mathbf{b} \in \mathcal{X}, \|\mathbf{a} - \mathbf{b}\|_2 \leq \gamma} (k(\mathbf{a}, \mathbf{a}) + k(\mathbf{b}, \mathbf{b}) - 2k(\mathbf{a}, \mathbf{b}))$. If $k(\cdot, \cdot)$ is continuous, then for any $\gamma > 0$, $f_{\mathcal{H}}(\gamma)$ is finite, and SVM is $(2\mathcal{N}(\gamma/2, \mathcal{X}, \|\cdot\|_2), \sqrt{f_{\mathcal{H}}(\gamma)/c})$ robust.

Example 6 (Lasso) *Let \mathcal{Z} be compact and the loss function be $l(\mathcal{A}_{\mathbf{s}}, z) = |z_{|y} - \mathcal{A}_{\mathbf{s}}(z_{|x})|$. Lasso (Tibshirani, 1996), which is the following regression formulation:*

$$\min_w : \frac{1}{n} \sum_{i=1}^n (s_{i|y} - w^\top s_{i|x})^2 + c\|w\|_1, \quad (6)$$

is $(\mathcal{N}(\gamma/2, \mathcal{Z}, \|\cdot\|_\infty), (Y(\mathbf{s})/c + 1)\gamma)$ -robust for all $\gamma > 0$, where $Y(\mathbf{s}) \triangleq \frac{1}{n} \sum_{i=1}^n s_{i|y}^2$.

Example 7 (Feed-forward Neural Networks) *Let \mathcal{Z} be compact and the loss function be $l(\mathcal{A}_{\mathbf{s}}, z) = |z_{|y} - \mathcal{A}_{\mathbf{s}}(z_{|x})|$. Consider the d -layer neural network (trained on \mathbf{s}), which is the following predicting rule given an input $x \in \mathcal{X}$*

$$\begin{aligned} x^0 &:= z_{|x} \\ \forall v = 1, \dots, d-1 : \quad & x_i^v := \sigma\left(\sum_{j=1}^{N_{v-1}} w_{ij}^{v-1} x_j^{v-1}\right); \quad i = 1, \dots, N_v; \\ \mathcal{A}_{\mathbf{s}}(x) &:= \sigma\left(\sum_{j=1}^{N_{d-1}} w_j^{d-1} x_j^{d-1}\right); \end{aligned}$$

If there exists α and β such that the d -layer neural network satisfying that $|\sigma(a) - \sigma(b)| \leq \beta|a - b|$, and $\sum_{j=1}^{N_v} |w_{ij}^v| \leq \alpha$ for all v, i , then it is $(\mathcal{N}(\gamma/2, \mathcal{Z}, \|\cdot\|_\infty), \alpha^d \beta^d \gamma)$ -robust, for all $\gamma > 0$.

We remark that in Example 7, the number of hidden units in each layer has no effect on the robustness of the algorithm and consequently the bound on the testing error. This indeed agrees with Bartlett (1998), where the author showed (using a different approach based on fat-shattering dimension) that for neural networks, the weight plays a more important role than the number of hidden units.

The next example considers an unsupervised learning algorithm, namely the principal component analysis algorithm. We show that it is robust if the sample space is *bounded*. This does not contradict with the well known fact that the principal component analysis is sensitive to outliers which are far away from the origin.

Example 8 (Principal Component Analysis (PCA)) Let $\mathcal{Z} \subset \mathbb{R}^m$ be such that $\max_{z \in \mathcal{Z}} \|z\|_2 \leq B$. If the loss function is $l((w_1, \dots, w_d), z) = \sum_{k=1}^d (w_k^\top z)^2$, then finding the first d principal components, which solves the following optimization problem over d vectors $w_1, \dots, w_d \in \mathbb{R}^m$,

$$\begin{aligned} \text{Maximize: } & \sum_{i=1}^n \sum_{k=1}^d (w_k^\top s_i)^2 \\ \text{Subject to: } & \|w_k\|_2 = 1, \quad k = 1, \dots, d; \\ & w_i^\top w_j = 0, \quad i \neq j. \end{aligned}$$

is $(\mathcal{N}(\gamma/2, \mathcal{Z}, \|\cdot\|_2), 2d\gamma B)$ -robust.

The last example is large-margin classification, which is a generalization of Example 1. We need the following standard definition (e.g., Bartlett, 1998) of the distance of a point to a classification rule.

Definition 12 Fix a metric ρ of \mathcal{X} . Given a classification rule Δ and $x \in \mathcal{X}$, the distance of x to Δ is

$$\mathcal{D}(x, \Delta) \triangleq \inf\{c \geq 0 \mid \exists x' \in \mathcal{X} : \rho(x, x') \leq c, \Delta(x) \neq \Delta(x')\}.$$

A large margin classifier is a classification rule such that most of the training samples are “far away” from the classification boundary.

Example 9 (Large-margin classifier) If there exist γ and \hat{n} such that

$$\sum_{i=1}^n \mathbf{1}(\mathcal{D}(s_{i|x}, \mathcal{A}_s) > \gamma) \geq \hat{n},$$

then algorithm \mathcal{A} is $(2\mathcal{N}(\gamma/2, \mathcal{X}, \rho), 0, \hat{n})$ pseudo robust, provided that $\mathcal{N}(\gamma/2, \mathcal{X}, \rho) < \infty$.

Proof: Set $\hat{\mathbf{s}}$ as

$$\hat{\mathbf{s}} \triangleq \{s_i \in \mathbf{s} \mid \mathcal{D}(s_i, \mathcal{A}_s) > \gamma\}.$$

And let $c_1, \dots, c_{\mathcal{N}(\gamma/2, \mathcal{X}, \rho)}$ be a $\gamma/2$ cover of \mathcal{X} . Thus, we can partition \mathcal{Z} to $2\mathcal{N}(\gamma/2, \mathcal{X}, \rho)$ subsets $\{C_i\}$, such that if

$$z_1, z_2 \in C_i; \implies y_1 = y_2; \ \& \ \rho(x_1, x_2) \leq \gamma.$$

This implies that:

$$z_1 \in \hat{\mathbf{s}}, z_1, z_2 \in C_i; \implies y_1 = y_2; \ \mathcal{A}_s(x_1) = \mathcal{A}_s(x_2); \implies l(\mathcal{A}_s, z_1) = l(\mathcal{A}_s, z_2).$$

By definition, \mathcal{A} is $(2\mathcal{N}(\gamma/2, \mathcal{X}, \rho), 0, \hat{n})$ pseudo robust. ■

Note that by taking ρ to be the Euclidean norm, and letting $\hat{n} = n$, we recover Example 1.

6 Necessity of Robustness

Thus far we have considered finite sample generalization bounds of robust algorithms. We now turn to asymptotic analysis, i.e., we are given an increasing set of training samples $\mathbf{s} = (s_1, s_2, \dots)$ and are tested on an increasing set of testing samples $\mathbf{t} = (t_1, t_2, \dots)$. We use $\mathbf{s}(n)$ and $\mathbf{t}(n)$ to denote the first n elements of training samples and testing samples respectively. For succinctness, we let $\mathcal{L}(\cdot, \cdot)$ to be the average loss given a set of samples, i.e., for $h \in \mathcal{H}$,

$$\mathcal{L}(h, \mathbf{t}(n)) \equiv \frac{1}{n} \sum_{i=1}^n l(h, t_i).$$

We show in this section that robustness is an essential property of successful learning. In particular, a (weaker) notion of robustness characterizes generalizability, i.e., a learning algorithm generalizes if and only if it is weakly robust. To make this precise, we define the notion of generalizability and weak robustness first.

Definition 13 1. A learning algorithm \mathcal{A} generalizes w.r.t. \mathbf{s} if

$$\limsup_n \left\{ \mathbb{E}_t \left(l(\mathcal{A}_{\mathbf{s}(n)}, t) \right) - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) \right\} \leq 0.$$

2. A learning algorithm \mathcal{A} generalize w.p. 1 if it generalize w.r.t. almost every \mathbf{s} .

We remark that the proposed notion of generalizability differs slightly from the standard one in the sense that the latter requires that the empirical risk and the expected risk converges in mean, while the proposed notion requires convergence w.p.1. It is straightforward that the proposed notion implies the standard one.

Definition 14 1. A learning algorithm \mathcal{A} is weakly robust w.r.t \mathbf{s} if there exists a sequence of $\{\mathcal{D}_n \subseteq \mathcal{Z}^n\}$ such that $\Pr(\mathbf{t}(n) \in \mathcal{D}_n) \rightarrow 1$, and

$$\limsup_n \left\{ \max_{\hat{\mathbf{s}}(n) \in \mathcal{D}_n} [\mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \hat{\mathbf{s}}(n)) - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n))] \right\} \leq 0.$$

2. A learning algorithm \mathcal{A} is a.s. weakly robust if it is robust w.r.t. almost every \mathbf{s} .

We briefly comment on the definition of weak robustness. Recall that the definition of robustness requires that the sample space can be partitioned into disjoint subsets such that if a testing sample belongs to the same partitioning set of a training sample, then they have similar loss. Weak robustness generalizes such notion by considering the average loss of testing samples and training samples. That is, if for a large (in the probabilistic sense) subset of \mathcal{Z}^n , the testing error is close to the training error, then the algorithm is weakly robust. It is easy to see, by Breteganolle-Huber-Carol lemma, that if for any fixed $\epsilon > 0$ there exists K such that \mathcal{A} is (K, ϵ) robust, then \mathcal{A} is weakly robust.

We now establish the main result of this section: weak robustness and generalizability are equivalent.

Theorem 15 An algorithm \mathcal{A} generalizes w.r.t. \mathbf{s} if and only if it is weakly robust w.r.t. \mathbf{s} .

Proof: We prove the sufficiency of weak robustness first. When \mathcal{A} is weakly robust w.r.t. \mathbf{s} , by definition there exists $\{D_n\}$ such that for any $\delta, \epsilon > 0$, there exists $N(\delta, \epsilon)$ such that for all $n > N(\delta, \epsilon)$, $\Pr(\mathbf{t}(n) \in D_n) > 1 - \delta$, and

$$\sup_{\hat{\mathbf{s}}(n) \in D_n} \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \hat{\mathbf{s}}(n)) - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) < \epsilon. \quad (7)$$

Therefore, the following holds for any $n > N(\delta, \epsilon)$,

$$\begin{aligned} & \mathbb{E}_{\mathbf{t}}(l(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{t})) - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) \\ &= \mathbb{E}_{\mathbf{t}(n)}(\mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{t}(n))) - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) \\ &= \Pr(\mathbf{t}(n) \notin D_n) \mathbb{E}(\mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{t}(n)) | \mathbf{t}(n) \notin D_n) + \Pr(\mathbf{t}(n) \in D_n) \mathbb{E}(\mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{t}(n)) | \mathbf{t}(n) \in D_n) \\ &\quad - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) \\ &\leq \delta M + \sup_{\hat{\mathbf{s}}(n) \in D_n} \{\mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \hat{\mathbf{s}}(n)) - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n))\} \leq \delta M + \epsilon. \end{aligned}$$

Here, the first equality holds since $\mathbf{t}(n)$ are i.i.d., and the second equality holds by conditional expectation. The inequalities hold due to the assumption that the loss function is upper bounded by M , as well as (7).

We thus conclude that the algorithm \mathcal{A} generalizes for \mathbf{s} , because ϵ and δ can be arbitrary.

Now we turn to the necessity of weak robustness. First, we establish the following lemma.

Lemma 16 Given \mathbf{s} , if algorithm \mathcal{A} is not weakly robust w.r.t. \mathbf{s} , then there exists $\epsilon^*, \delta^* > 0$ such that the following holds for infinitely many n ,

$$\Pr(\mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{t}(n)) \geq \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) + \epsilon^*) \geq \delta^*. \quad (8)$$

Proof: We prove the lemma by contradiction. Assume that such ϵ^* and δ^* do not exist. Let $\epsilon_v = \delta_v = 1/v$ for $v = 1, 2, \dots$, then there exists a non-decreasing sequence $\{N(v)\}_{v=1}^{\infty}$ such that for all v , if $n \geq N(v)$ then $\Pr(\mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{t}(n)) \geq \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) + \epsilon_v) < \delta_v$. For each n , define the following set:

$$\mathcal{D}_n^v \triangleq \{\hat{\mathbf{s}}(n) | \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \hat{\mathbf{s}}(n)) - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) < \epsilon_v\}.$$

Thus, for $n \geq N(v)$ we have

$$\Pr(\mathbf{t}(n) \in \mathcal{D}_n^v) = 1 - \Pr\left(\mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{t}(n)) \geq \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) + \epsilon_v\right) > 1 - \delta_v.$$

For $n \geq N(1)$, define $\mathcal{D}_n \triangleq \mathcal{D}_n^{v(n)}$, where: $v(n) \triangleq \max(v | N(t) \leq n; v \leq n)$. Thus for all $n \geq N(1)$ we have that $\Pr(\mathbf{t}(n) \in \mathcal{D}_n) > 1 - \delta_{v(n)}$ and $\sup_{\hat{\mathbf{s}}(n) \in \mathcal{D}_n} \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \hat{\mathbf{s}}(n)) - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) < \epsilon_{v(n)}$. Note that $v(n) \uparrow \infty$, it follows that $\delta_{v(n)} \rightarrow 0$ and $\epsilon_{v(n)} \rightarrow 0$. Therefore, $\Pr(\mathbf{t}(n) \in \mathcal{D}_n) \rightarrow 1$, and

$$\limsup_{n \rightarrow \infty} \left\{ \sup_{\hat{\mathbf{s}}(n) \in \mathcal{D}_n} \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \hat{\mathbf{s}}(n)) - \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) \right\} \leq 0.$$

That is, \mathcal{A} is weakly robust w.r.t. \mathbf{s} , which is a desired contradiction. \blacksquare

We now prove the necessity of weak robustness. Recall that $l(\cdot, \cdot)$ is uniformly bounded. Thus by Hoeffding's inequality we have that for any ϵ and δ , there exists n^* such that for any $n > n^*$, with probability at least $1 - \delta$, we have $\left| \frac{1}{n} \sum_{i=1}^n l(\mathcal{A}_{\mathbf{s}(n)}, t_i) - \mathbb{E}_t(l(\mathcal{A}_{\mathbf{s}(n)}, t)) \right| \leq \epsilon$. This implies that

$$\mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{t}(n)) - \mathbb{E}_t l(\mathcal{A}_{\mathbf{s}(n)}, t) \xrightarrow{\Pr} 0. \quad (9)$$

Since algorithm \mathbb{A} is not weakly robust, Lemma 16 implies that (8) holds for infinitely many n . This, combined with Equation (9) implies that for infinitely many n ,

$$\mathbb{E}_t l(\mathcal{A}_{\mathbf{s}(n)}, t) \geq \mathcal{L}(\mathcal{A}_{\mathbf{s}(n)}, \mathbf{s}(n)) + \frac{\epsilon^*}{2},$$

which means that \mathcal{A} does not generalize. Thus, the necessity of weak robustness is established. \blacksquare

Theorem 15 immediately leads to the following corollary.

Corollary 17 *An algorithm \mathcal{A} generalizes w.p. 1 if and only if it is a.s. weakly robust.*

7 Discussion

In this paper we investigated the generalization of learning algorithm based on their robustness: the property that if a testing sample is “similar” to a training sample, then its loss is close to the training error. This provides a novel approach, different from complexity or stability arguments, in studying the performance of learning algorithms. We further showed that a weak notion of robustness characterizes generalizability, which implies that robustness is the fundamental property that makes learning algorithms work.

Before concluding the paper, we outline several directions for future research.

- *Adaptive partition:* In Definition 2 when the notion of robustness was introduced, we required that the partitioning of \mathcal{Z} into K sets is *fixed*. That is, regardless of the training sample set, we partition \mathcal{Z} into the same K sets. A natural and interesting question is what if such fixed partition does not exist, while instead we can only partition \mathcal{Z} into K sets *adaptively*, i.e., for different training set we will have a different partitioning of \mathcal{Z} . Adaptive partition setup can be used to study algorithms such as k-NN. Our current proof technique does not straightforwardly extend to such a setup, and we would like to understand whether a meaningful generalization bound under this weaker notion of robustness can be obtained.
- *Mismatched datasets:* One advantage of algorithmic robustness framework is the ability to handle non-standard learning setups. For example, in Section 3.2 we derived generalization bounds for quantile loss. A problem of the same essence is the *mismatched datasets*, also called as *domain adaption*, see Ben-David et al. (2007), Mansour et al. (2009) and reference therein. Here the training samples are generated according to a distribution slightly different from that of the testing samples, e.g., the two distributions may have a small K-L divergence. We conjecture that in this case a generalization bound similar to Theorem 3 would be possible, with an extra term depending on the magnitude of the difference of the two distributions.
- *Outlier removal:* One possible reason that the training samples is generated differently from the testing sample is corruption by outliers. It is often the case that the training sample set is corrupted by some outliers. In addition, algorithms designed to be outlier resistant abound in the literature (Huber, 1981; Rousseeuw & Leroy, 1987). The robust framework may provide a novel approach in studying both the generalization ability and the outlier resistant property of these algorithms. In particular, the results reported in Section 3.2 can serve as a starting point of future research in this direction.

- *Consistency*: We addressed in this paper the relationship between robustness and generalizability. An equally important feature of learning algorithms is *consistency*: the property that a learning algorithm guarantees to recover the global optimal solution as the size of the training set increases. While it is straightforward that if an algorithm minimizes the empirical error asymptotically and also generalizes (or equivalently is *weakly robust*), then it is consistent, much less is known for a necessary condition for an algorithm to be consistent. It is certainly interesting to investigate the relationship between consistency and robustness, and in particular whether robustness is necessary for consistency, at least for algorithms that asymptotically minimize the empirical error.
- *Other robust algorithms*: The proposed robust approach considers a general learning setup. However, except for PCA, the algorithms investigated in Section 5 are in the supervised learning setting. One natural extension is to investigate other robust unsupervised and semi-supervised learning algorithms. One difficulty is that compared to supervised learning case, the analysis of unsupervised/semi-supervised learning algorithms can be challenging, due to the fact that many of them are random iterative algorithms (e.g., k-means).

Acknowledgement

We thank Constantine Caramanis and the anonymous reviewers for their insightful comments. The research of Huan Xu was partially supported by DTRA grant HDTRA1-08-0029. Shie Mannor was supported by a Horev Fellowship, by the European Union under a Marie-Curie Reintegration Grant and by the Israel Science Foundation (contract 890015).

References

- Alon, N., Ben-David, S., Cesa-Bianchi, N., & Haussler, D. (1997). Scale-sensitive dimension, uniform convergence, and learnability. *Journal of the ACM*, 44, 615–631.
- Bartlett, P. L. (1998). The sample complexity of pattern classification with neural networks: The size of the weight is more important than the size of the network. *IEEE Transactions on Information Theory*, 44, 525–536.
- Bartlett, P. L., Bousquet, O., & Mendelson, S. (2005). Local Rademacher complexities. *The Annals of Statistics*, 33, 1497–1537.
- Bartlett, P. L., & Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3, 463–482.
- Ben-David, S., Blitzer, J., Crammer, K., & Pereira, F. (2007). Analysis of representations for domain adaptation. *Advances in Neural Information Processing Systems 19*.
- Ben-Tal, A., & Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, 23, 769–805.
- Ben-Tal, A., & Nemirovski, A. (1999). Robust solutions of uncertain linear programs. *Operations Research Letters*, 25, 1–13.
- Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations Research*, 52, 35–53.
- Bhattacharyya, C., Pannagadatta, K. S., & Smola, A. J. (2004). A second order cone programming formulation for classifying missing data. *Advances in Neural Information Processing Systems (NIPS17)*. Cambridge, MA: MIT Press.
- Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research*, 2, 499–526.
- Cortes, C., & Vapnik, V. N. (1995). Support vector networks. *Machine Learning*, 20, 1–25.
- Devroye, L., Györfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition*. Springer, New York.
- Devroye, L., & Wagner, T. (1979a). Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions of Information Theory*, 25, 202–207.

- Devroye, L., & Wagner, T. (1979b). Distribution-free performance bounds for potential function rules. *IEEE Transactions of Information Theory*, *25*, 601–604.
- Evgeniou, T., Pontil, M., & Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Large Margin Classifiers* (pp. 171–203). Cambridge, MA: MIT Press.
- Globerson, A., & Roweis, S. (2006). Nightmare at test time: Robust learning by feature deletion. *Proceedings of the 23rd International Conference on Machine Learning* (pp. 353–360). New York, NY, USA: ACM Press.
- Huber, P. J. (1981). *Robust statistics*. John Wiley & Sons, New York.
- Mansour, Y., Mohri, M., & Rostamizadeh, A. (2009). Domain adaptation: Learning bounds and algorithms. *Proceedings of The 22nd Annual Conference on Learning Theory*.
- McDiarmid, C. (1989). On the method of bounded differences. *Surveys in Combinatorics* (pp. 148–188).
- Mukherjee, S., Niyogi, P., Poggio, T., & Rifkin, R. (2006). Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, *25*, 161–193.
- Poggio, T., Rifkin, R., Mukherjee, S., & Niyogi, P. (2004). General conditions for predictivity in learning theory. *Nature*, *428*, 419–422.
- Rousseeuw, P. J., & Leroy, A. M. (1987). *Robust regression and outlier detection*. John Wiley & Sons, New York.
- Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. MIT Press.
- Shalev-Shwartz, S., Shamir, O., Srebro, N., & Sridharan, K. (2009). Learnability and stability in the general learning setting. *Proceedings of 22nd Annual Conference of Learning Theory*.
- Shivaswamy, P. K., Bhattacharyya, C., & Smola, A. J. (2006). Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, *7*, 1283–1314.
- Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, *58*, 267–288.
- van der Vaart, A. W., & Wellner, J. A. (2000). *Weak convergence and empirical processes*. Springer-Verlag, New York.
- Vapnik, V. N., & Chervonenkis, A. (1974). *Theory of pattern recognition*. Nauka, Moscow.
- Vapnik, V. N., & Chervonenkis, A. (1991). The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, *1*, 260–284.
- Xu, H., Caramanis, C., & Mannor, S. (2009). Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, *10*, 1485–1510.
- Xu, H., Caramanis, C., & Mannor, S. (2010). Robust regression and Lasso. To appear in *IEEE Transactions on Information Theory*.
- Xu, H., & Mannor, S. (2010). Robustness and generalization. ArXiv: 1005.2243.

Learning to create is as hard as learning to appreciate

David Xiao *

Abstract

We explore the relationship between a natural notion of unsupervised learning studied by Kearns et al. (STOC '94), which we call here “learning to create” (LTC), and the standard PAC model of Valiant (CACM '84), which is a form of supervised learning and can be thought of as a formalization of “learning to appreciate”. Our main theorem states that “if learning to appreciate is hard, then so is learning to create”. That is, we prove that if PAC learning with respect to efficiently samplable input distributions is hard, then solving the LTC problem is also hard. We also investigate ways in which our result are tight.

1 Introduction

The **P** vs. **NP** question is often cast in the intuitively appealing language of “creativity” and whether “creativity can be automated” (see *e.g.* the survey of Wigderson [28]). To explain this view, one often uses as an analogy a great artist, say Beethoven, who produced widely appreciated works of music. We model the process of deciding whether a piece of music is pleasing by an efficient circuit f_{music} . Then the process of creating pleasing music amounts to finding a satisfying assignment to f_{music} , while appreciating music only requires evaluating f_{music} on a given input.¹ Therefore, if **P** = **NP**, one can automate the task of composing pleasing music because there would be an efficient algorithm that found pleasing pieces of music (*i.e.* the satisfying assignments of f_{music}).

The above analogy is not the only way one can view creativity through a computational lens. In this paper we explore the question of automating creativity from a learning-theoretic point of view. We will explore two models of learning that correspond to “learning to appreciate” and “learning to create”. Both the studied models are standard: the first is the PAC model of Valiant [27], while the second is a form of unsupervised learning, which we call “learning to create” (LTC), whose complexity-theoretic study was initiated by Kearns et al. [16].

PAC learning is a “label prediction” problem, and in particular is a form of *supervised* learning. In the PAC model, the learning algorithm is given examples labelled according to a hidden function f and is supposed to learn how to label new examples as f would. For example, we might try to learn how a particular person Alice appreciates music. In this case, we model Alice’s taste by a function $f_{\text{music}}^{\text{Alice}}$ that takes input a piece of music and outputs whether or not Alice finds it pleasing. The learning task would be, given a set of examples of music each labelled $f_{\text{music}}^{\text{Alice}}$, to output a hypothesis that labels new pieces of music the same way as $f_{\text{music}}^{\text{Alice}}$ would. Of course, Alice’s taste may be very different from another person Bob, so the same learning algorithm should successfully learn f_{music}^p for *all* persons p , given examples labelled according to f_{music}^p .

LTC is a “pattern reconstruction” problem, and in particular is a form of *unsupervised* learning. In the LTC model, the learning algorithm is given many unlabelled examples drawn from a hidden distribution D , and is supposed to construct a circuit that generates new examples that are

*LRI, Université Paris-Sud, dxiao@lri.fr

¹Of course the analogy is not entirely accurate since we believe that **P** \neq **NP** while, presumably, Beethoven was bound by the Extended Church-Turing Hypothesis and could not solve **NP**-hard problems. But let us ignore this detail and suppose that Beethoven’s creativity was indeed the result of solving an **NP**-hard problem.

distributed close to D . One can think of D as say generating a piece of music as Beethoven would. Of course, the Beethoven’s distribution of music $D_{\text{Beethoven}}$ is very different from, say, $D_{\text{John Lennon}}$, and the learning algorithm should learn using examples how to produce music according to one style or the other.

In this paper, when we refer to the PAC or LTC problem, we mean solving these problems for “complete” concept classes (unless we specifically say otherwise). For instance, we study whether it is possible to PAC learn all labellings computable by $\text{SIZE}(n^2)$ circuits, and whether it is possible to solve LTC for the class of distributions samplable by $\text{SIZE}(n^2)$ circuits.

Our first result says roughly that if PAC is hard, then so is LTC.

Theorem 1.1 (LTC is as hard as PAC learning, informal). *If the PAC learning problem with respect to efficiently samplable input distributions cannot be solved by a polynomial-time algorithm, then the LTC problem cannot be solved by a polynomial-time algorithm.*

This theorem holds even for the more stringent requirement of agnostic learning. In addition to our analogy about learning to create vs. learning to appreciate, one can also interpret our result as saying that “unsupervised learning is as hard as supervised learning” in the context of these particular models.

One may ask whether [Theorem 1.1](#) can be strengthened to say that *for every concept class \mathcal{F}* , if it is hard to PAC learn \mathcal{F} , then it is also hard to solve LTC for the class \mathcal{F} .² This is a stronger statement than [Theorem 1.1](#), since, as we will see, the proof of [Theorem 1.1](#) will take a class \mathcal{F} that is hard in the PAC model and transform it into a (more complex) class \mathcal{F}' that is hard in the LTC model. We show that this stronger statement is false by exhibiting concrete concept classes for which it does not hold.

Theorem 1.2 (PAC vs. LTC for specific concept classes, informal). *Under standard cryptographic assumptions, there exist concept classes for which PAC learning (even with respect to the uniform input distribution) is hard while LTC is easy.*

Another weakness in [Theorem 1.1](#) is that it considers only PAC learning with respect to efficiently samplable input distributions. In general, PAC learning allows the examples given to the learning algorithm to be generated from *any* distribution; [Theorem 1.1](#) adds the restriction that the examples must be generated from a distribution that can be sampled by, say, a $\text{SIZE}(n^2)$ circuit.

In our last result, we study whether [Theorem 1.1](#) can be generalized to encompass PAC learning with respect to *unsamplable* input distributions. We show this is unlikely: we exhibit an oracle relative to which LTC is easy (and therefore PAC learning is easy for all efficiently samplable input distributions), but there exist functions that are hard to learn with respect to an unsamplable distribution.

Theorem 1.3 (Separating LTC and PAC learning unsamplable input distributions). *There exists an oracle \mathcal{O} relative to which solving LTC for $\text{SIZE}^{\mathcal{O}}(n^2)$ is easy, while there is a function $f \in \text{SIZE}^{\mathcal{O}}(n^2)$ that is an efficiently computable function that is hard to PAC learn on an unsamplable input distribution.*

1.1 Relation to previous work

Hardness of learning. It is widely believed that both the PAC and LTC problems are hard. Both problems can be proven hard if one assumes the existence of cryptography [[27](#), [17](#)] or the weaker assumption that zero knowledge is non-trivial [[1](#)] (see also [Corollary 2.11](#)). However, to the best of our knowledge, prior to this work there were no results establishing a relationship *between* the hardness of PAC learning and the LTC problem.

Previous work on complexity of LTC. The computational complexity of PAC learning has been studied extensively since its first appearance [[27](#), [15](#), [22](#), [17](#), [4](#)]. The complexity of LTC was first studied in [[16](#)] but overall is less well-understood. One question about LTC that has received some attention is its relation to a related notion of “learning to evaluate probabilities”: given samples as in the LTC problem, construct a hypothesis $h : \{0, 1\}^n \rightarrow [0, 1]$ such that $h(x) = \Pr[D = x]$, *i.e.* the hypothesis evaluates the probability that x is drawn from D . It is known that under reasonable assumptions, there is a concrete concept class for which “learning to evaluate probabilities” is hard and LTC is easy [[16](#)], while there is also a concept class for which “learning to evaluate probabilities” is easy yet LTC is hard [[20](#)].

²Since LTC deals with classes of functions with multi-bit outputs while PAC deals with classes of functions with single-bit outputs, it must be clarified how we obtain both single- and multi-bit functions from a single class \mathcal{F} . We defer this discussion to [Section 4](#).

Complexity of learning via complete problems. Much of the literature comparing different learning models has focused on *concrete* problems. In this kind of comparison, one exhibits a single concept class that is learnable in one model but not in another (under some reasonable complexity assumption). This has led to valuable insights into the complexity of various models, including the power of membership oracles [5], faulty vs. perfect membership oracles [25, 6], and the aforementioned difference between learning to evaluate probabilities and LTC [16, 20]. Furthermore, this kind of comparison is perhaps the most reasonable when comparing models where one model is obviously more complex than the other, and the goal is to show that this relationship is in some sense strict (e.g. [5], where clearly having a membership oracle makes the learning task easier).

On the other hand, this approach has the drawback that in some cases it can lead to conflicting evidence, as in the case of learning to evaluate probabilities and LTC, where looking at one concept class suggests that the learning in the first model is harder than in the second model, but looking at a different concept class suggests the opposite.

A different approach to understanding the complexity of the learning model is to examine a *complete problem* for the model. Namely, if we consider the problem of learning a “complete” concept class (e.g. $\text{SIZE}(n^2)$ circuits) in learning model M , then the existence of any hard-to-learn concept class for M would imply that learning $\text{SIZE}(n^2)$ is hard. This approach was explored in Applebaum et al. [1], Xiao [29, 30] to understand the complexity of PAC learning relative to the complexity of NP , auxiliary-input one-way functions [21], and zero knowledge. In this paper we apply this approach to the complete problems for PAC learning and for LTC.

PAC learning with respect to efficient distributions. We already noted that [Theorem 1.1](#) only relates LTC to PAC learning or agnostic learning with respect to efficiently samplable input distributions. We believe that this is a reasonable restriction: according to the strong Church-Turing thesis, all physical phenomena can be explained by efficient (polynomial-time) computation. Therefore, one can suppose that from whatever source one obtains the examples to be learned, if one can suppose that they are i.i.d. samples from a distribution then one may as well suppose that this distribution is polynomial-time samplable. Furthermore, [Theorem 1.3](#) says that no relativizing reduction can strengthen [Theorem 1.1](#) to include PAC learning with respect to unsamplable distributions.

1.2 Our techniques

1.2.1 Proving [Theorem 1.1](#)

Cryptography using circuits. Our [Theorem 1.1](#) is proven using a variation of standard cryptographic techniques. Standard cryptography is based on *uniform* cryptographic primitives: for example, one-way functions or pseudo-random functions that are computable using one Turing Machine for all input lengths. This is because in order to use these primitives, one needs an efficient way to compute them for any desired input size. On the other hand, it is often required that these primitives are hard-to-break even for non-uniform families of polynomial-size circuits, since it may happen that the adversary has some side-information about the cryptosystem that is best modelled as non-uniform advice.

Because we are looking at things from a learning-theoretic point of view, we consider analogues of one-way functions and pseudorandom functions that are computable by circuits (with non-uniform advice), but which are only required to be secure against uniform adversaries. Such primitives, which we call one-way circuits or pseudorandom circuits in this paper, were studied in the context of zero knowledge, first in Ostrovsky and Wigderson [21] and later in Vadhan [26], who used a slightly different definition. (They were called *auxiliary-input one-way functions* in these contexts.) The connection between one-way circuits and learning theory (also linking learning theory to zero knowledge) was explored in Applebaum et al. [1], Xiao [29, 30]. The main fact about one-way circuits we use to prove [Theorem 1.1](#) is that if one-way circuits exist, then the LTC problem is hard ([Corollary 2.11](#)).

Circuit agnostic learning. Another ingredient in the proof of [Theorem 1.1](#) is a variation of the standard PAC/agnostic learning problem that we call circuit agnostic learning, implicit in [1] and made explicit in [30] (see [Definition 3.1](#)). Whereas in the standard PAC/agnostic learning models, the learning algorithm only receives a collection of labelled examples, in the circuit agnostic learning model the learning algorithm is given a circuit that samples the distribution of labelled examples. Notice that this does not trivialize the problem: knowing a circuit that *generates* labelled examples does not necessarily reveal how to *label* examples. See [Section 3.1](#) for more discussion.

We know that if one-way circuits exist, then PAC learning is hard [1], but the converse is unknown to hold, namely it is unknown whether the hardness of PAC learning implies that one-way circuits exist (and in fact, it was shown in [29] that this converse cannot be proven by relativizing

techniques). However, the converse *does* hold if we consider circuit agnostic learning: if PAC learning is hard, then the circuit agnostic learning problem is hard (see Lemma 3.2). This is the second main ingredient that we use in the proof of Theorem 1.1.

Proof idea behind Theorem 1.1. Combining the above two tools, the proof idea for Theorem 1.1 is to use a LTC algorithm to solve PAC learning with respect to efficiently samplable distributions as follows. First, obtain a set of labelled examples, and use the LTC algorithm on this set of labelled examples to obtain a circuit C that samples the distribution from which these labelled examples were drawn. Note that such a circuit exists because we are promised that the PAC learning instance is on an input distribution that is efficiently samplable. This effectively transforms the original PAC learning problem into an instance of the circuit agnostic learning problem. Next, combining our two theorems about one-way circuits (Corollary 2.11) and circuit agnostic learning (Lemma 3.2), it is possible to use the LTC algorithm to solve this circuit agnostic learning problem, which in turn solves the original PAC learning problem as well.

1.2.2 Understanding Theorem 1.3

Oracle separations. Theorems such as Theorem 1.3 are called *oracle separations*, and have long been used in theoretical computer science to “separate” various classes. Baker et al. [2] showed an oracle separation between \mathbf{P} and \mathbf{NP} and Impagliazzo and Rudich [14] showed an oracle separation between one-way permutations and the intuitively harder task of key exchange. More recently, such oracle separations were also used in learning theory to separate PAC learning from the hardness of zero knowledge [29].

The motivation behind such oracle separations is as follows. There are very few unconditional separations in theoretical computer science (and almost non-existent when going beyond weak complexity classes such as $\mathbf{AC0}$). On the other hand, many if not most complexity results are proved using relativizing techniques (for example black-box reductions and diagonalization). Therefore by proving an oracle separation between classes A and B , one shows that in order to prove A reduces to B , one would need to come up with a non-relativizing technique. This arguably attests to the difficulty of the task.³ In this spirit, we interpret Theorem 1.3 to mean that strengthening Theorem 1.1 to encompass PAC learning even with respect to unsamplable distributions would require new non-relativizing techniques.

2 Preliminaries

2.1 Notation and basic lemmas

If X is a probability distribution, then let $\text{supp}(X)$ denote the support of X , *i.e.* the values that X takes on with positive probability. For two distributions X_1, X_2 , the total variation distance (or statistical distance) is defined as $\Delta(X_1, X_2) = \frac{1}{2} \sum_x |\Pr[X_1 = x] - \Pr[X_2 = x]|$, where the sum is taken over all x in the supports of X_1, X_2 . We let U_n denote the uniform distribution $\{0, 1\}^n$. For finite sets S , we will sometimes abuse notation and let S also stand for the uniform distribution over S . For a circuit $C : \{0, 1\}^m \rightarrow \{0, 1\}^n$, we say that C samples a distribution X if $C(U_m) = X$. We let $\text{SIZE}(q(n))$ denote the set of (functions computable by) circuits of size $q(n)$, and we let $\text{SIZE}^{\mathcal{O}}(q(n))$ denote the same where the circuits are allowed oracle gates \mathcal{O} at unit cost.

We will use the following lemma of Borel-Cantelli, which says that if a countable sequence of events each have small probability of occurring, then the probability that an infinite number of them occurs is 0.

Theorem 2.1 (Borel-Cantelli lemma). *Let $\{E_n\}_{n \in \mathbb{N}}$ be a sequence of events. Suppose $\sum_{n=1}^{\infty} \Pr[E_n]$ exists and is finite. Then $\Pr[\exists I \subseteq \mathbb{N}, |I| = \infty, \forall n \in I, E_n \text{ occurs}] = 0$.*

2.2 PAC learning (or learning to appreciate)

Define the learning error of a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ with respect to a distribution (X, Y) over $\{0, 1\}^{n+1}$ to be $\text{err}((X, Y), f) = \Pr_{X, Y}[f(X) \neq Y]$. For a class of functions \mathcal{F} , define $\text{err}((X, Y), \mathcal{F}) = \min_{f \in \mathcal{F}} \text{err}((X, Y), f)$.

Definition 2.2 (PAC Learning). An algorithm A PAC learns the concept class \mathcal{F} if the following holds for every $n \in \mathbb{N}, \varepsilon > 0, f \in \mathcal{F}$, and every distribution X over $\{0, 1\}^n$. Given access to an example oracle that generates labelled examples according to $(X, f(X))$, A produces with success probability $\geq 1 - 2^{-n}$ an ε -good hypothesis h (represented as a circuit), namely $\text{err}((X, f(X)), h) \leq \varepsilon$. Furthermore, A runs in time $\text{poly}(n, 1/\varepsilon)$.

³Of course, such a result does *not* imply that the task is impossible. Indeed, many of the most surprising results bypass oracle separations or other notions of separation (for example [19, 24, 3]).

Definition 2.3. We say that A learns \mathcal{F} *w.r.t. efficient distributions* if the PAC learning guarantee is only required to hold for all $X = C(U_m)$, where $m = \text{poly}(n)$ and C is a polynomial-size circuit.

Worst possible error is 1/2: we will assume that A always outputs a hypothesis h such that $\text{err}((X, f(X)), h) \leq 1/2$. One can assure that this occurs with probability $\geq 1 - 2^{-n}$ by checking whether the majority of labels output by h agrees with the majority on the examples drawn from the oracle, and if they disagree outputting $1 - h$ instead of h .

A word on padding: In this paper we use $\text{SIZE}(n^2)$, the class of functions computable by size n^2 circuits, as a complete concept class. This class is complete for functions computable by polynomial-size circuits because of a padding argument. For example, in order to learn circuits of size n^c , it suffices to pad an example x of length n to $x0^{n^{c/2}-n}$ which has length $n' = n^{c/2}$, and then running the learner for $\text{SIZE}(n^2)$. This same kind of padding works for the LTC setting defined below.

Agnostic learning: we will also work with an even more demanding notion of learning, called *agnostic learning*, defined in Kearns et al. [17], where the examples may not be labelled according to any fixed function. Here, the goal is to obtain a hypothesis that performs (almost) as well as the best hypothesis in a concept class.

Definition 2.4 (Agnostic Learning). A procedure A agnostically learns the concept class \mathcal{F} if the following holds for every $n \in \mathbb{N}, \varepsilon > 0$, and every distribution (X, Y) over $\{0, 1\}^{n+1}$. Given access to an example oracle that generates labelled examples according to (X, Y) , A produces with success probability $\geq 1 - 2^{-n}$ an ε -good hypothesis h (represented as a circuit), namely $\text{err}((X, Y), h) \leq \text{err}((X, Y), \mathcal{F}) + \varepsilon$. Furthermore, A runs in time $\text{poly}(n, 1/\varepsilon)$.

Definition 2.5. We say that A agnostically learns \mathcal{F} *w.r.t. efficient distributions* if the agnostic learning guarantee holds for all $(X, Y) = C(U_m)$, where $m = \text{poly}(n)$ and C is a polynomial-size circuit.

2.3 Learning to create

Definition 2.6 (LTC). A procedure A solves LTC for the concept class \mathcal{F} if the following holds for every $n \in \mathbb{N}, \varepsilon > 0$, and $f \in \mathcal{F}$, where $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$. Given access to an oracle that generates samples according to $f(U_m)$, A outputs with probability $1 - \varepsilon$ an ε -close hypothesis $h : \{0, 1\}^{m'} \rightarrow \{0, 1\}^n$ (represented as a circuit), namely $\Delta(f(U_m), h(U_{m'})) \leq \varepsilon$.⁴ Furthermore, A runs in time $\text{poly}(n, 1/\varepsilon)$.

The accuracy of the hypotheses: our definition of PAC learning requires a strong notion of accuracy: for an example oracle (X, Y) , we require the hypothesis h to satisfy $\text{err}((X, Y), h) \leq 1/\text{poly}(n)$. In the PAC model we know one may apply Boosting [23, 7, 8] to show that “strong PAC learning” is equivalent to “weak PAC learning”, where the hypothesis is only required to satisfy $\text{err}((X, Y), h) \leq \frac{1}{2} - 1/\text{poly}(n)$. In contrast, there is no known equivalent boosting technique for the LTC problem, so we must acknowledge that our definition that $\Delta(f(U_m), h(U_{m'})) \leq 1/\text{poly}(n)$ is indeed a strong requirement, and this is necessary for our results.

2.4 Cryptography using circuits

We assume the reader is familiar with the standard notions of one-way functions and pseudorandom functions/permutations, and refer to [9] for further details.

Definition 2.7. Pseudorandom circuits (PRC) exist if for every efficient uniform algorithm D , there exists an infinite collection W of functions where for every $f \in W, f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$, f is computable by a circuit of size $s(n) = \text{poly}(n)$ and it holds that

$$\left| \Pr_{D, k \stackrel{R}{\leftarrow} U_n} [D^{f_k}(f, 1^s) = 1] - \Pr_{D, \phi} [D^\phi(f, 1^s) = 1] \right| \leq s^{-\omega(1)} \quad (2.1)$$

where f is passed to D as a circuit, $f_k = f(k, \cdot)$ and ϕ is a truly random function from $\{0, 1\}^n \rightarrow \{0, 1\}$.

f is a (uniform) pseudorandom permutation (PRP) if $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, f_k is a permutation for all k , f is computable by a uniform Turing Machine, and Equation 2.1 holds for all efficient D .

⁴Kearns et al. [16] defined closeness using KL divergence. We use statistical distance as this is sufficient in most applications and simplifies our presentation. All of our results also hold for KL divergence with appropriate (but qualitatively equivalent) scaling of parameters.

Definition 2.8. Distributional one-way circuits (DOWC) exist if for every efficient algorithm I , there exists a polynomial $p(s)$ and an infinite collection W of functions where for every $f \in W$, $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, f is computable by a circuit of size $s(n) = \text{poly}(n)$ and it holds that

$$\Delta((x, f(x)), (I(f, y), y \mid y = f(x))) > 1/p(s)$$

over random choice of $x \leftarrow_{\mathbb{R}} U_n$ and the random coins of I , and f is given to I as a circuit.

Our definition is based on [21] (although we require the inverter to *distributionally* invert the circuit, *i.e.* it must output a nearly-uniform preimage rather than an arbitrary preimage). One-wayness and pseudorandom functions are known to be equivalent for the uniform case [11, 10, 13], and the reductions establishing this extend immediately to the non-uniform case.

Theorem 2.9 ([11, 10, 13]). *DOWC exist if and only if PRC exist.*

2.5 Solving LTC implies inverting circuits

It was shown in Kearns et al. [16] that one can use an algorithm solving LTC in order to distinguish PRP from truly random functions. By looking at their proof, we observe that it also applies to PRC.

Theorem 2.10 (Kearns et al. [16]). *If LTC is efficiently solvable for the class $\text{SIZE}(n^2)$, then PRC do not exist, i.e. there is a polynomial-time algorithm that distinguishes any efficient circuit from a truly random function.*

Corollary 2.11 (Follows from Theorem 2.10 and Theorem 2.9). *If LTC is efficiently solvable for the class $\text{SIZE}(n^2)$, then no family of circuits is distributionally one-way, i.e. there is a polynomial-time algorithm that distributionally inverts any polynomial-size circuit.*

3 Solving LTC implies solving agnostic learning w.r.t. efficient distributions

The idea of our proof of Theorem 1.1 is that we can use an LTC solver to learn the circuit that samples the distribution of labelled examples. This makes the PAC learning problem easier because we now have a circuit generating labelled examples (rather than just a set of labelled examples), and we show that the LTC solver can also be used to solve this relaxed PAC learning problem.

3.1 Circuit agnostic learning

To prove Theorem 1.1, we use a tool called “circuit agnostic learning”. In standard notions of learning, the learning algorithm is given access only to examples drawn from the distribution (X, Y) . One can also ask what happens when the learning algorithm gets access to a circuit that samples from the distribution (X, Y) . For the setting of agnostic learning, we call this relaxed (and potentially easier) problem *circuit agnostic learning*:

Definition 3.1. A procedure A circuit-agnostic-learns a concept class \mathcal{F} if on input circuit C of size s sampling a distribution (X, Y) over $\{0, 1\}^{n+1}$, A outputs a hypothesis h such that

$$\text{err}((X, Y), h) \leq \text{err}((X, Y), \mathcal{F}) + \varepsilon$$

and A runs in time $\text{poly}(s, 1/\varepsilon)$.

At first glance it might seem that this model is trivially easy because the learning algorithm has access to C , which allows the learning algorithm to generate labelled examples by himself and may allow the learning algorithm to create a good hypothesis. However the problem remains non-trivial because C generates an example and its label *simultaneously*, while the problem the learning algorithm must solve is to compute the label on an example given as input.⁵

The following lemma implicitly was proved in [1] and explicitly appears in Xiao [30]

Lemma 3.2 ([1] (see also [30], Lemma 3.5.1)). *If there is an efficient algorithm that distributionally inverts all polynomial-size circuits, then there is an algorithm running in time $\text{poly}(n, 1/\varepsilon)$ that circuit-agnostically-learns $\text{SIZE}(n^2)$.*

⁵To see a concrete example of a class for which circuit-agnostic learning is hard, consider the concept class in Section 4.2. For this concept class the standard PAC learning problem and the circuit agnostic learning problem are equivalent, since after $O(\log n)$ samples in the standard PAC model one can obtain the modulus N , which using Algorithm 4.8 allows one to construct a circuit sampling the input distribution $(U_n, f_N(U_n))$. This means that the Quadratic Residuosity assumption implies that the circuit agnostic learning problem for this concept class is hard.

3.2 Proof of Theorem 1.1

Theorem 3.3 (Theorem 1.1, formal). *If there exists a polynomial-time algorithm A_{LTC} that solves LTC for the class $\text{SIZE}(n^2)$, then there exists a polynomial-time algorithm A_{Agn} that solves agnostic learning with respect to the concept class $\text{SIZE}(n^2)$ and with respect to all distributions samplable by $\text{SIZE}(n^2)$ circuits.*

Proof of Theorem 3.3. By hypothesis there exists a polynomial-time algorithm A_{LTC} that solves LTC for the class $\text{SIZE}(n^2)$. By Corollary 2.11 it follows that there is a polynomial-time algorithm I that distributionally inverts all circuits. By Lemma 3.2 there is a polynomial-time algorithm $A_{\text{CircLearn}}$ that circuit-agnostically-learns $\text{SIZE}(n^2)$.

Defining the agnostic learning algorithm: the algorithm A_{Agn} that learns agnostically w.r.t. efficiently samplable distributions does the following. By padding, we may assume that the input distribution X, Y is sampled by a circuit of size n^2 . First, A_{Agn} obtains enough samples $(x_1, y_1), \dots, (x_{t(n)}, y_{t(n)})$ from the example oracle to run A_{LTC} with error parameter $\varepsilon/2$. Let C be its output. Run $A_{\text{CircLearn}}$ on C with error $\varepsilon/2$, and let h be the output of $A_{\text{CircLearn}}$. Output h .

Analyzing A_{Agn} : since A_{LTC} solves LTC for the concept class $\text{SIZE}(n^2)$, we get with probability $1 - 2^{-n}$ a hypothesis $C : \{0, 1\}^m \rightarrow \{0, 1\}^n$ such that $\Delta(C(U_m), (X, Y)) \leq \varepsilon/2$. Letting (X', Y') be the distribution samples by C , this implies that $\Delta((X', Y'), (X, Y)) \leq \varepsilon/2$. Since $A_{\text{CircLearn}}$ solves $\text{CircLearn}^{\text{SIZE}(n^2)}$, it follows that with probability $1 - 2^{-n}$, the output hypothesis h satisfies $\text{err}((X', Y'), h) \leq \varepsilon/2$. Together, it follows that $\text{err}((X, Y), h) \leq \varepsilon$. ■

Remark 3.4. All of the ingredients used in the proof of Theorem 3.3 relativize, and therefore the statement of Theorem 3.3 also relativizes. Namely, relative to any oracle \mathcal{O} , if solving LTC for the class $\text{SIZE}^{\mathcal{O}}(n^2)$ is easy, then PAC learning $\text{SIZE}^{\mathcal{O}}(n^2)$ with respect to input distributions samplable by $\text{SIZE}^{\mathcal{O}}(\text{poly}(n))$ circuits is easy.

4 LTC and PAC learning for concrete classes

In this section we show that, in contrast to Theorem 1.1, if one studies concrete concept classes that are not complete, then it is possible that PAC learning is harder than LTC.

Because PAC learning deals with single-bit output function while LTC deals with multi-bit output functions, in order to compare the two models for concrete concept classes we use two different ways to obtain both single- and multi-bit output functions from a single concept class:

1. Direct products: let \mathcal{F} be a class of single-bit output functions. Then we compare PAC learning the class \mathcal{F} to solving LTC for the class \mathcal{F}^ℓ where each function $f \in \mathcal{F}^\ell$ maps $\{0, 1\}^n \rightarrow \{0, 1\}^\ell$ and can be decomposed as $f(x) = (f_1(x), \dots, f_\ell(x))$ where each $f_i \in \mathcal{F}$. Here, the number of copies $\ell(n)$ satisfies $\omega(\log n) \leq \ell(n) \leq \text{poly}(n)$.
2. Generating labelled examples: let \mathcal{F} be a class of single-bit output functions and let \mathcal{D} be a class of distributions that are efficiently samplable. Then we compare PAC learning \mathcal{F} with respect to input distributions in \mathcal{D} to solving LTC for the class of distributions of the form $(X, f(X))$ where $X \in \mathcal{D}$ and $f \in \mathcal{F}$.

To motivate the above notions, the direct product notion is natural when thinking of \mathcal{F} as being a syntactic complexity class, such as DNF formulas or AC0 circuits. Thus, in the PAC model the function to be learned has complexity \mathcal{F} , and similarly in the LTC problem each bit of output in the output distribution has complexity \mathcal{F} .

The “generating labelled examples” notion is motivated by the proof of Theorem 1.1. In the proof of Theorem 1.1, the first step is to apply the LTC algorithm to produce a circuit that generates (approximately) the distribution of labelled examples. By considering this notion, we will see that the proof of Theorem 1.1 does not immediately generalize to hold for concrete concept classes.

Since we have no unconditional lower bounds for polynomial-time computation (which would be necessary to show that polynomial-time algorithms cannot solve PAC or LTC), all of the following results are conditional, *i.e.* they assume that some (standard) computational problem is hard.

4.1 Direct product

Proposition 4.1. *Assuming one-way functions exist, then there exists a concept class \mathcal{F} that is hard to learn in the PAC model but such that solving LTC for the class \mathcal{F}^ℓ is easy.*

Proof. Since we assume one-way functions exist, therefore [11, 10, 18] implies that there exists a (uniform) pseudorandom permutation of the form $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ (we use the shorthand $f_k(x) = f(k, x)$) such that f_k is a permutation, and for all efficient distinguishers D ,

$$|\Pr_{\phi, D}[D^\phi(1^n) = 1] - \Pr_{k, D}[D^{f_k}(1^n) = 1]| \leq \varepsilon(n)$$

where the distinguishing advantage $\varepsilon(n) = n^{-\omega(1)}$ is negligible. We make the following claim, which says that there exists an infinite collection of keys K such that f_k is hard-to-compute for all $k \in K$:

Lemma 4.2. *Let $\{f_k\}_{k \in \{0, 1\}^*}$ be a collection of pseudorandom permutations with distinguishing advantage $\varepsilon(n)$. Then there exists an infinite set $K = \{k_n\}_{n \in \mathbb{N}}$ such that $\forall k \in K, n = |k|$, it holds for all efficient algorithms A that for large enough n ,*

$$\Pr_A[A^{f_{k_n}} = h \text{ and } \text{err}((U_n, f_{k_n}(U_n)), h) \leq 1/2 - 2\sqrt{\varepsilon(n)}] \leq n^2 \sqrt{\varepsilon(n)}$$

We will prove this lemma shortly, first we use it to define \mathcal{F} and prove [Proposition 4.1](#).

Defining \mathcal{F} : let $K = \{k_n\}_{n \in \mathbb{N}}$ be the set of hard keys defined by [Lemma 4.2](#). Let $f_{k_n}(x)_i$ denote the i 'th bit of $f_{k_n}(x)$. We define

$$\mathcal{F} = \bigcup_{n \in \mathbb{N}} \{g_i : \{0, 1\}^n \rightarrow \{0, 1\}, g_i(x) = f_{k_n}(x)_i \mid i \in [n]\}$$

Claim 4.3. *PAC learning \mathcal{F} is hard.*

This follows from [Lemma 4.2](#): if there were an algorithm that PAC learns \mathcal{F} , then it could in particular be used to compute f_{k_n} for all n : given oracle access to f_{k_n} , one can simulate example oracles $(U_n, g_i(U_n))$ for all $i \in [n]$, and using the PAC learning algorithm for \mathcal{F} with error $1/n^2$ one could obtain with high probability hypotheses h_i such that $\Pr[h_i(U_n) = g_i(U_n)] \geq 1 - 1/n^2$. Letting $h = (h_1, \dots, h_n)$, we see that $\text{err}((U_n, f_{k_n}(U_n)), h) \leq 1/n$, which contradicts [Lemma 4.2](#) since $1/n \ll 1/2 - 2\sqrt{\varepsilon(n)}$.

Claim 4.4. *Solving LTC for the class \mathcal{F}^ℓ is easy.*

Fix any function $(g_{i_1}, \dots, g_{i_\ell}) \in \mathcal{F}^\ell$. Since f_{k_n} is a permutation, $f_{k_n}(U_n)$ is uniform. Therefore, $g_{i_p}(U_n)$ and $g_{i_q}(U_n)$ are independent uniform bits if $i_p \neq i_q$, and they are always equal if $i_p = i_q$.

We now describe an algorithm that solves LTC for the class \mathcal{F}^ℓ . Let D be the distribution to be learned, $D = (g_{i_1}(r), \dots, g_{i_\ell}(r) \mid r \leftarrow_{\mathbb{R}} U_n)$.

1. Initialize a graph G on n vertices to be the complete graph, where the vertices are labelled $1, \dots, n$.
2. Repeat the following $t = n \log \binom{n}{2}$ times. Sample $x \leftarrow_{\mathbb{R}} D$, and for every pair $u, v \in [n]$ such that $x_u \neq x_v$, remove the edge (u, v) from G .
3. The output hypothesis h does the following: for each connected component of G , sample a random bit. Output x where x_u equals the bit of the connected component containing u .

We claim that with all but probably 2^{-n} , the distribution sampled by h will be *exactly* the distribution generated by $(g_{i_1}, \dots, g_{i_\ell})$. The only time that this hypothesis will be different is if there is some pair u, v such that $i_u \neq i_v$ and yet, for all examples x that the algorithm draws, it holds that $x_{i_u} = x_{i_v}$. Since for each sample this happens independently with probability $1/2$, and since there are $t = n \log \binom{n}{2}$ samples, by a union bound over all edges this happens with probability at most $\binom{n}{2} 2^{-n \log \binom{n}{2}} \leq 2^{-n}$. ■

Proof of [Lemma 4.2](#). For an algorithm A and a key k of length n , let

$$p_{A,k} = \Pr_A[A^{f_k} = h \text{ and } \text{err}((U_n, f_k(U_n)), h) \leq 1/2 - 2\sqrt{\varepsilon(n)}]$$

where the probability is only over the random coins of A . It must hold that:

Claim 4.5. *For all sufficiently large n , $\Pr_{k \leftarrow_{\mathbb{R}} \{0, 1\}^n} [p_{A,k} > n^2 \sqrt{\varepsilon(n)}] \leq 1/n^2$*

This claim holds because otherwise one could use A to break the pseudorandomness of f by the following distinguisher D . First D runs A to obtain h . Then, D queries its oracle on a new uniform $x \leftarrow_{\mathbb{R}} U_n$; let b be the oracle's response. D accepts if $b = h(x)$ and rejects otherwise. It is easy to compute $\Pr_{\phi, D}[D^\phi(1^n) = 1] \leq \frac{1}{2} + p(n)2^{-n}$ where $p(n) = \text{poly}(n)$ is the maximum number of queries made by A . On the other hand, if [Claim 4.5](#) does not hold, then $\Pr_{k, D}[D^{f_k}(1^n) = 1] \geq \frac{1}{2} + 2\varepsilon(n)$. We can assume *w.l.o.g.* that $\varepsilon \geq 2^{-n/2}$ ([Lemma 4.2](#) only gets weaker if we increase ε), therefore this gives a distinguishing probability $2\varepsilon - p(n)2^{-n} \gg \varepsilon(n)$, contradicting the pseudorandomness of f_k .

Define $p_A = \Pr_{k_1, k_2, \dots}[\text{For infinitely many } n, p_{A, k_n} > n^2 \sqrt{\varepsilon(n)}]$ where $k_n \leftarrow_{\mathbb{R}} \{0, 1\}^n$. Since the series $\sum_{n=1}^{\infty} 1/n^2 < \infty$, applying [Theorem 2.1](#) and [Claim 4.5](#) implies that $p_A = 0$. Since there is a countable number of algorithms A , this implies

$$\Pr_{k_1, k_2, \dots} [\exists A, \text{ For infinitely many } n, p_{A, k_n} > n^2 \sqrt{\varepsilon(n)}] \leq \sum_A p_A = 0$$

Therefore, a random choice of K will satisfy the conclusion of [Lemma 4.2](#) with probability 1. ■

4.2 Generating labelled examples

Our result for this model is based on the hardness of quadratic residuosity over Blum integers. We say that $N = pq$ is a Blum integer of length n if p, q are prime, $\lceil \log N \rceil = n$, $n - \lceil \log p \rceil \leq 2$, $n - \lceil \log q \rceil \leq 2$ and $p \equiv q \equiv 3 \pmod{4}$. We say that x is a quadratic residue mod a if $\exists y \in \mathbb{Z}_N$ such that $x = y^2 \pmod{a}$ for $a \in \mathbb{N}$. The *Legendre symbol* $(\frac{x}{p})$ is equal to 0 if $x = 0 \pmod{p}$, it is equal to 1 if x is a quadratic residue mod p and -1 if x is a quadratic non-residue mod p . The *Jacobi symbol* is defined $(\frac{x}{N}) = (\frac{x}{p})(\frac{x}{q})$. It is possible to efficiently compute the Jacobi symbol using Euclid's algorithm. It is known that for a Blum integer N , $(\frac{-1}{N}) = 1$ but -1 is *not* a quadratic residue mod N .

Let $\text{QR}(N, x) = 1$ if $x = y^2 \pmod{N}$ and 0 otherwise, and write $\text{QR}_N(x) = \text{QR}(N, x)$. The hardness of quadratic residuosity over Blum integers says that there is no polynomial time algorithm that evaluates $\text{QR}_N(x)$ given a Blum integer N and $x \in \mathbb{Z}_N$.^{6 7}

Proposition 4.6. *Assuming that Quadratic Residuosity is hard over Blum integers, there is a concept class \mathcal{F} for which PAC learning with respect to the uniform distribution is hard while solving LTC for distributions $(U_n, f(U_n))$ where $f \in \mathcal{F}$ is easy.*

Proof. Define the functions $f_N : \mathbb{Z}_N \times [\lceil \log N \rceil] \times \{0, 1\} \rightarrow \{0, 1\}$ where

$$f_N(x, i, b) = \begin{cases} \text{QR}_N(x) & b = 0 \\ N_i & b = 1 \end{cases}$$

where N_i denotes the i 'th bit of N . Let n the input length of f_N and let $\mathcal{F} = \{f_N \mid N \text{ is a Blum integer}\}$.

Claim 4.7. *PAC learning \mathcal{F} is hard for the uniform distribution.*

Suppose we have a PAC learning algorithm A for \mathcal{F} (it even suffices if A only works for uniformly distributed inputs). Given N , one can simulate an example oracle for $(U_n, f_N(U_n))$ as follows:

Algorithm 4.8 (Sampling from $(U_n, f_N(U_n))$):

1. Pick $x' \leftarrow_{\mathbb{R}} \mathbb{Z}_N, i \leftarrow_{\mathbb{R}} [\lceil \log N \rceil], b \leftarrow_{\mathbb{R}} \{0, 1\}$.
2. If $b = 1$ then output $((x', i, b), N_i)$, otherwise compute the Jacobi symbol $(\frac{x'}{N})$.
3. If $(\frac{x'}{N}) \neq 1$, then output $((x', i, b), 0)$.
4. If $(\frac{x'}{N}) = 1$, then sample $r \leftarrow_{\mathbb{R}} \mathbb{Z}_N, a \leftarrow_{\mathbb{R}} \{-1, 1\}$, and output $((ar^2, i, b), \frac{1+a}{2})$.

This simulated example oracle is identical to a true example oracle for f_N .

⁶Here, the fact that the factorization N is unknown to the algorithm is necessary to ensure hardness. Indeed, it is possible to compute $\text{QR}_N(x)$ using an efficient circuit that has the factorization p, q as advice. This is, for instance, why the class \mathcal{F} defined in [Proposition 4.6](#) is efficiently computable.

⁷This example can also be phrased in terms of the generic assumption that trapdoor permutations exist, but the presentation using Quadratic Residuosity is simpler.

Using a PAC learner to define an algorithm A' solving quadratic residuosity: on input (x, N) , A' first checks if $(\frac{x}{N}) \neq 1$, and if so outputs 0. Otherwise, use [Algorithm 4.8](#) to simulate an example oracle for $(U_n, f_N(U_n))$, then run A on this example oracle to obtain a hypothesis h . Finally, A' picks a random $r \leftarrow_{\mathbb{R}} \mathbb{Z}_N, i \leftarrow_{\mathbb{R}} [n]$ and outputs $h(xr^2, i, 0)$.

We claim that A' solves quadratic residuosity. Let S_1 be the set of quadratic residues in \mathbb{Z}_N , and let S_{-1} be the set of quadratic non-residues $y \in \mathbb{Z}_N$ with Jacobi symbol $(\frac{y}{N}) = 1$. Observe that $\Pr_{y \leftarrow_{\mathbb{R}} \mathbb{Z}_N}[y \in S_1] = \Pr_{y \leftarrow_{\mathbb{R}} \mathbb{Z}_N}[y \in S_{-1}] = 1/4 + o(1)$. Therefore, for all $a \in \{-1, 1\}$, it holds over uniform i that

$$\text{err}(((S_a, i, 0), f_N(S_a, i, 0)), h) \leq (8 + o(1))\text{err}((U_n, f_N(U_n)), h) \leq 9\varepsilon$$

for large enough n . Since for $a \in \{-1, 1\}$ and every $x \in S_a$, the variable $xr^2 \bmod N$ for random r is distributed uniformly in S_a , this implies that A' outputs $\text{QR}_N(x)$ correctly with probability $1 - 9\varepsilon - 2^{-n}$.

Claim 4.9. *Solving LTC for $(U_n, f_N(U_n))$ for $f_N \in \mathcal{F}$ is easy.*

After seeing $O((\log N)^2) = \text{poly}(n)$ samples, with all but negligible probability, N is revealed. Given N , one can sample from $(U_n, f_N(U_n))$ using [Algorithm 4.8](#). ■

5 PAC learning unsamplable distributions

Our construction of \mathcal{O} will be randomized: we will select \mathcal{R} from a distribution of oracles and show that with high probability that the oracle $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$ satisfies [Theorem 1.3](#). The distribution will be as follows:

Definition 5.1. On input length n , let $\mathcal{R} : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ be chosen as follows: select $z \leftarrow_{\mathbb{R}} \{0, 1\}^n$. Pick the set S_z from the following distribution: for each $x \in \{0, 1\}^n$, put x into S_z with probability $2^{-n/2}$. Then, for each $x \in S_z$, let $\mathcal{R}_z(x) = \mathcal{R}(z, x) = 1$ with probability $1/2$ and 0 otherwise. For all $z' \neq z$, let $\mathcal{R}_{z'}(x) = 0$ for all $x \in \{0, 1\}^n$.

Intuitively, for each input length we first pick a “hard instance” z , then we pick a “hard set” S_z that is a sparse random subset of $\{0, 1\}^n$ of size roughly $2^{n/2}$. We then define \mathcal{R}_z to be a random function on S_z and 0 elsewhere, and also $\mathcal{R}_{z'}$ is identically zero for all $z' \neq z$. We remark that the definition of this oracle was also proposed by Impagliazzo [\[12\]](#), but as an alternative oracle for the main result of [\[29\]](#). It was not studied with respect to the question of this paper, and the analyses we provide below are new.

Proof of [Theorem 1.3](#). We will show that with overwhelming probability over choice of such \mathcal{R} , solving LTC relative to $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$ is easy, but PAC learning relative to $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$ with respect to unsamplable distributions is hard.

Lemma 5.2 (PAC learning unsamplable distribution is hard). *With probability 1 over \mathcal{R} , for all efficient algorithms A with oracle access to $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$, and for all but finitely many n , let $z \in \{0, 1\}^n$ be the hard instance on length n , then*

$$\Pr_A[A^{\mathcal{O}} \text{ given access to } (S_z, \mathcal{R}_z(S_z)) \text{ outputs } h \text{ s.t. } \text{err}((S_z, \mathcal{R}_z(S_z)), h) \leq \frac{1}{2} - n^{-\log n}] \leq n^{-\log n}$$

The proof of this is deferred to the full version. The intuition is that $\mathcal{R}_z(x)$ looks like a random bit: the only way an algorithm could predict $\mathcal{R}_z(x)$ is either if x was one of the examples it was given in the set of labelled examples, or if the learning algorithm finds the value z so that it can query the oracle at $\mathcal{R}_z(x)$. The first case is unlikely because $|S_z| \approx 2^{n/2}$, while the second is unlikely because z is chosen uniformly at random and therefore hard for the learning algorithm to find.

On the other hand, the following also holds:

Lemma 5.3 (Solving LTC easy). *There is an efficient $A^{\mathcal{O}}$ such that with probability 1 over the choice of \mathcal{R} where $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$, $A^{\mathcal{O}}$ solves LTC for the concept class $\text{SIZE}^{\mathcal{O}}(n^2)$.*

We sketch the proof of this lemma shortly. Together, these two lemmas imply [Theorem 1.3](#). Notice that we did not need to prove separately that S_z is unsamplable; this follows immediately since learning w.r.t. S_z is hard while learning w.r.t. efficiently samplable distributions is easy: [Remark 3.4](#) and [Lemma 5.3](#) imply that there is an efficient algorithm that solves PAC learning for the concept class $\text{SIZE}^{\mathcal{O}}(n^2)$ with respect to efficiently samplable distributions. ■

Proof of Lemma 5.3. The maximum likelihood approach: We will use a maximum likelihood approach to solve LTC. The maximum likelihood algorithm says that, given a sample $T = (x_1, \dots, x_t)$ that was obtained from one distribution out of a class of distributions \mathcal{D} , it suffices to pick the $D \in \mathcal{D}$ such that $\Pr[D^t = T]$ is maximized.

More formally, let $\text{ML}_{\mathcal{D}}(x_1, \dots, x_t) = \text{argmax}_{D \in \mathcal{D}} \{\Pr[D^t = (x_1, \dots, x_t)]\}$. The following holds:

Claim 5.4 (Folklore). *Fix \mathcal{D} of size $|\mathcal{D}| \leq 2^{\text{poly}(n)}$ and such that for every $D \in \mathcal{D}$ and every $x \in \text{supp}(D)$, $\Pr[D = x] \geq 2^{-\text{poly}(n)}$. Then for $t = \text{poly}(n, 1/\varepsilon)$ and for any $D \in \mathcal{D}$, it holds that:*

$$\Pr_{x_1, \dots, x_t \leftarrow RD} [\text{ML}_{\mathcal{D}}(x_1, \dots, x_t) = D' \wedge \Delta(D', D) > \varepsilon] \leq 2^{-n}$$

We defer a proof to the full version.

Let $\mathcal{D}^{\mathcal{O}}$ be the class of distributions sampled by circuits in $\text{SIZE}^{\mathcal{O}}(n^2)$. To solve LTC for $\text{SIZE}^{\mathcal{O}}(n^2)$, we would like to run the maximum likelihood approach over $\mathcal{D}^{\mathcal{O}}$. However, in order to calculate or even roughly approximate $\Pr[(D^{\mathcal{O}})^t = (x_1, \dots, x_t)]$ for distributions $D^{\mathcal{O}}$ that might be sampled by circuits including \mathcal{O} gates, one would need to know how $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$ behaves everywhere, and this requires querying \mathcal{R} exponentially many times.

Our approach is to still use the maximum likelihood approach, but rather than applying the approach using $\mathcal{D}^{\mathcal{O}}$ as the class of distributions, we apply it to a related class $\mathcal{D}'_q = \{\mathcal{D}'_{q,n}\}_{n \in \mathbb{N}}$ for which having a \mathbf{PSPACE} oracle is sufficient to calculate the maximum likelihood hypothesis.

Defining \mathcal{D}'_q using truncated oracles: $\mathcal{D}'_{q,n}$ will be the following class of distributions. For $z \in \{0, 1\}^n, S \subseteq \{0, 1\}^n$, define the function $R_n^{z,S} : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ to be $R_n^{z,S}(z, x) = 1$ if $x \in S$ and zero elsewhere. Note that $R_n^{z,S}$ can be concisely represented if $|S| = \text{poly}(n)$. The class of $q(n)$ -truncated oracles on length n is the following:

$$\mathcal{R}_{q,n} = \left\{ R = (R_1^{z_1, S_1}, \dots, R_n^{z_n, S_n}) \mid \forall i \in [n], z_i \in \{0, 1\}^i, S_i \subseteq \{0, 1\}^i, |S_i| \leq q(n) \right\}$$

A distribution D is in $\mathcal{D}'_{q,n}$ if there exists an $R \in \mathcal{R}_{q,n^2}$ and oracle circuit of size n^2 that has (in addition to AND, OR, and NOT gates) \mathbf{PSPACE} gates and R gates. Note that the circuit is allowed oracle gates only for a single $R \in \mathcal{R}_{q,n^2}$, or in other words it cannot have two different kind of oracle gates evaluating two different $R \neq R' \in \mathcal{R}_{\varepsilon, n^2}$. Also note that because we can explicitly represent S_i , each of these circuits is contained in $\text{SIZE}^{\mathbf{PSPACE}}(n^3(1+q))$.

The following straightforward claim says that, with a \mathbf{PSPACE} oracle, it is possible to efficiently evaluate the probability that $D \in \mathcal{D}'_{q,n}$ generates a particular sample:

Claim 5.5. *There exists an algorithm using a \mathbf{PSPACE} oracle that runs in time $\text{poly}(n, 1/\varepsilon)$ and computes $\text{ML}_{\mathcal{D}'_{q,n}}$.*

This follows from the simple fact that calculating $\Pr[D = x]$ for a distribution D that is samplable by a $\text{SIZE}^{\mathbf{PSPACE}}(\text{poly}(n))$ circuit can be efficiently done with a \mathbf{PSPACE} oracle. Since, as remarked above, $\mathcal{D}'_{q,n}$ can be sampled by $\text{SIZE}^{\mathbf{PSPACE}}(n^3(1+q))$ circuits, this suffices to build the algorithm in Claim 5.5.

It therefore remains to prove that, with high probability over the choice of function \mathcal{R} , the class $\mathcal{D}'_{q,n}$ is a good approximation for the class $\mathcal{D}^{\mathcal{O}} = \{\mathcal{D}_n^{\mathcal{O}}\}_{n \in \mathbb{N}}$ sampled by circuits in $\text{SIZE}^{\mathcal{O}}(n^2)$. We say that $\mathcal{D}_n^{\mathcal{O}}$ is ε -approximable by \mathcal{D}' if for every $D \in \mathcal{D}_n^{\mathcal{O}}$, there exists $D' \in \mathcal{D}'$ such that $\Delta(D, D') \leq \varepsilon$.

Lemma 5.6. *For all n, ε , let $q = 16n^9/\varepsilon^3$, then $\Pr_{\mathcal{R}}[\mathcal{D}_n^{\mathcal{O}} \text{ is } \varepsilon\text{-approximable by } \mathcal{D}'_{q,n}] > 1 - 2^{-n}$.*

We defer the proof of this lemma to the full version. We briefly sketch the intuition here: let $C \in \text{SIZE}^{\mathcal{O}}(n^2)$ be the circuit sampling D . Following an idea of [29], we prove that it is only necessary to know the queries that C makes to \mathcal{R} that are “heavy”, *i.e.* that occur with large probability. Then we can simply replace \mathcal{R} gates by a truth table that includes values for all the heavy queries. This modified circuit is a circuit in \mathcal{D}'_q , and we show that this modification does not change the behavior of the output distribution by much.

Next use Lemma 5.6 to prove the theorem.

The learning algorithm A_{LTC} . We now combine our claims to obtain the following algorithm:

Algorithm 5.7.

Algorithm A_{LTC} : input size n , error parameter ε .

1. Let $t = \text{poly}(n, 2/\varepsilon)$ be the appropriate polynomial to apply [Claim 5.4](#) with error $\varepsilon/2$. A_{LTC} draws t examples x_1, \dots, x_t from D .
2. Using the algorithm of [Claim 5.5](#), set $q = 16n^9(2t/\varepsilon)^3$ and compute $D' = \text{ML}_{\mathcal{D}'_{q,n}}(x_1, \dots, x_t)$. Output D' .

Proof of correctness: We prove that A_{LTC} indeed solves the LTC problem for $\text{SIZE}^{\mathcal{O}}(n^2)$. By [Theorem 2.1](#), it suffices to show that for all n , with probability $1 - 2^{-n}$ over the choice of \mathcal{R} , it holds for all $D \in \mathcal{D}_n^{\mathcal{O}}$ that

$$\Pr_{x_1, \dots, x_t \leftarrow_{\mathcal{R}} D} [A_{\text{LTC}}(x_1, \dots, x_t) = D' \quad \wedge \quad \Delta(D', D) > \varepsilon] \leq \varepsilon \tag{5.1}$$

(This error can be reduced to 2^{-n} by repeating A_{LTC} and taking the best hypothesis it output.) For $q = 16n^9(2t/\varepsilon)^3$, [Lemma 5.6](#) implies that with probability $1 - 2^{-n}$ over the choice of \mathcal{R} , $\mathcal{D}_n^{\mathcal{O}}$ is $(\varepsilon/2t)$ -approximable by $\mathcal{D}'_{q,n}$. In this case, for every $D \in \mathcal{D}_n^{\mathcal{O}}$, there exists $D' \in \mathcal{D}'_{q,n}$ such that by the triangle inequality it holds that $\Delta(D^t, (D')^t) \leq \varepsilon/2$. Therefore

$$\begin{aligned} \Pr_{x_1, \dots, x_t \leftarrow_{\mathcal{R}} D} [A_{\text{LTC}} = D' \wedge \Delta(D', D) > \varepsilon] &\leq \Pr_{x_1, \dots, x_t \leftarrow_{\mathcal{R}} D'} [A_{\text{LTC}} = D'' \wedge \Delta(D'', D) > \varepsilon] + \varepsilon/2 \\ &\leq \Pr_{x_1, \dots, x_t \leftarrow_{\mathcal{R}} D'} [A_{\text{LTC}} = D'' \wedge \Delta(D'', D') > \varepsilon - \varepsilon/(2t)] + \varepsilon/2 \\ &\leq 2^{-n} + \varepsilon/2 \leq \varepsilon \end{aligned}$$

where penultimate inequality follows from [Claim 5.4](#), since $D' \in \mathcal{D}'_{q,n}$ and A_{LTC} evaluates $\text{ML}_{\mathcal{D}'_{q,n}}$ (the conditions of the hypothesis are satisfied because D is samplable by a polynomial-size oracle circuit). This proves [Equation 5.1](#). Furthermore, observe that [Claim 5.5](#) implies that A_{LTC} runs in polynomial time using a $(\mathcal{R}, \text{PSPACE})$ oracle. ■

6 Acknowledgements

The author would like to thank the anonymous referees for their helpful comments.

References

- [1] B. Applebaum, B. Barak, and D. Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Proc. FOCS '08*, pages 211–220, 2008.
- [2] T. Baker, J. Gill, and R. Solovay. Relativizations of the $\mathcal{P} = ?\mathcal{NP}$ question. *SIAM Journal on Computing*, 4(4):431–442, 1975. doi: 10.1137/0204037. URL <http://link.aip.org/link/?SMJ/4/431/1>.
- [3] B. Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd FOCS*, pages 106–115. IEEE, 2001.
- [4] A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In *CRYPTO '93*, pages 278–291, 1993. ISBN 3-540-57766-1.
- [5] V. Feldman. On the power of membership queries in agnostic learning. *J. Mach. Learn. Res.*, 10:163–182, 2009. ISSN 1532-4435.
- [6] V. Feldman and S. Shah. Separating models of learning with faulty teachers. *Theor. Comput. Sci.*, 410(19):1903–1912, 2009. ISSN 0304-3975. doi: <http://dx.doi.org/10.1016/j.tcs.2009.01.017>.
- [7] Y. Freund. Boosting a weak learning algorithm by majority. In *Proc. COLT '90*, pages 202–216, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. ISBN 1-55860-146-5.
- [8] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Comp. and Sys. Sci.*, 55(1):119–139, 1997.
- [9] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001. Earlier version available on <http://www.wisdom.weizmann.ac.il/~oded/frag.html>.
- [10] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. ISSN 0004-5411. Preliminary version in FOCS' 84.
- [11] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. of Com.*, 28(4):1364–1396, 1999. Preliminary versions appeared in STOC' 89 and STOC' 90.

- [12] R. Impagliazzo. Private communication, 2009.
- [13] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Proc. 30th FOCS*, pages 230–235, 1989.
- [14] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC '89*, pages 44–61. ACM, 1989. ISBN 0-89791-307-8. doi: <http://doi.acm.org/10.1145/73007.73012>.
- [15] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proc. STOC '89*, pages 433–444, New York, NY, USA, 1989. ACM. ISBN 0-89791-307-8. doi: <http://doi.acm.org/10.1145/73007.73049>.
- [16] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 273–282, New York, NY, USA, 1994. ACM. ISBN 0-89791-663-8. doi: <http://doi.acm.org/10.1145/195058.195155>.
- [17] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning. In *COLT '92*, pages 341–352, 1992. ISBN 0-89791-497-X. doi: <http://doi.acm.org/10.1145/130385.130424>.
- [18] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. of Com.*, 17(2):373–386, 1988. Preliminary version in *STOC' 86*.
- [19] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proc. 31st FOCS*, pages 2–10. IEEE, 1990.
- [20] M. Naor. Evaluation may be easier than generation (extended abstract). In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 74–83, New York, NY, USA, 1996. ACM. ISBN 0-89791-785-5. doi: <http://doi.acm.org/10.1145/237814.237833>.
- [21] R. Ostrovsky and A. Wigderson. One-way functions are essential for non-trivial zero-knowledge. Technical Report TR-93-073, International Computer Science Institute, Berkeley, CA, Nov. 1993. Preliminary version in *Proc. 2nd Israeli Symp. on Theory of Computing and Systems*, 1993, pp. 3–17.
- [22] L. Pitt and M. K. Warmuth. Prediction-preserving reducibility. *J. Comput. Syst. Sci.*, 41(3): 430–467, 1990. ISSN 0022-0000. doi: [http://dx.doi.org/10.1016/0022-0000\(90\)90028-J](http://dx.doi.org/10.1016/0022-0000(90)90028-J).
- [23] R. Schapire. The strength of weak learnability. *Proc. FOCS '89*, pages 28–33, 1989. doi: <http://doi.ieeecomputersociety.org/10.1109/SFCS.1989.63451>.
- [24] A. Shamir. $Ip = pspace$. *J. ACM*, 39(4):869–877, 1992. ISSN 0004-5411. doi: <http://doi.acm.org/10.1145/146585.146609>.
- [25] H.-U. Simon. How many missing answers can be tolerated by query learners? In *STACS '02: Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 384–395, London, UK, 2002. Springer-Verlag. ISBN 3-540-43283-3.
- [26] S. P. Vadhan. An unconditional study of computational zero knowledge. *FOCS '04*, pages 176–185, 2004. ISSN 0272-5428. doi: <http://doi.ieeecomputersociety.org/10.1109/FOCS.2004.13>.
- [27] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. ISSN 0001-0782. doi: <http://doi.acm.org/10.1145/1968.1972>.
- [28] A. Wigderson. \mathbf{P} , \mathbf{NP} , and mathematics - a computational complexity perspective. In *Proceedings of the ICM '06*, volume 1, pages 665–712, Zurich, Switzerland, 2006. EMS Publishing House.
- [29] D. Xiao. On basing $\mathbf{ZK} \neq \mathbf{BPP}$ on the hardness of PAC learning. In *In Proc. CCC '09*, pages 304–315, 2009.
- [30] D. Xiao. *New Perspectives on the Complexity of Computational Learning, and Other Problems in Theoretical Computer Science*. PhD thesis, Princeton University, 2009.

Author Index

Abernethy, Jacob	318	Minsker, Stas	420
Ackerman, Margareta	270	Moore, Andrew W.	295
Agarwal, Alekh.....	28	Munos, Remi	41, 477
Akavia, Adi.....	381	Neu, Gergely.....	231
Alon, Noga.....	27	Reidenbach, Daniel.....	194
Audibert, Jean-Yves	41	Rigollet, Philippe.....	54
Awasthi, Pranjali.....	359	Ryabko, Daniil.....	119
Balcan, Maria Florina	282	Sarkar, Purnamrita	295
Barbosa, Marconi.....	451	Schapiro, Robert E.....	308
Belkin, Mikhail	407	Scholkopf, Bernhard.....	464
Ben-David, Shai.....	270	Shalev-Shwartz, Shai.....	14, 218, 441
Blum, Avrim.....	359	Shamir, Ohad.....	218, 441
Bubeck, Sebastien.....	41, 477	Sheffet, Or	359
Caramanis, Constantine	490	Shie Mannor,	503
Case, John.....	181	Singer, Yoram.....	14, 257
Cesa-Bianchi, Nicolo	218, 320	Sinha, Kaushik.....	407
Chakrabarti, Deepayan.....	295	Sridharan, Karthik	1, 346, 441
Crammer, Koby	80, 168	Steudel, Bastian	464
de Rooij, Steven.....	106	Streeter, Matthew.....	244
Dekel, Ofer	28, 346	Szepesvari, Csaba	231
Duchi, John C.	14, 257	Takemura, Akimichi.....	67
Even-Dar, Eyal	168	Tewari, Ambuj.....	1, 14
Freydenberger, Dominik D.....	194	Valiant, Leslie G.....	155
Gavinsky, Dmitry	207	Vaughan, Jennifer Wortman.....	155, 168
Gentile, Claudio.....	320, 346	Vitale, Fabio	320
Geulen, Sascha.....	132	Výocking, Berthold	132
Golovin, Daniel.....	333	Wan, Andrew.....	368
Gottlieb, Lee-Ad	433	Warmuth, Manfred K.	93, 144, 314
Grunwald, Peter	106	Wasserman, Larry.....	394
Gupta, Anupam.....	394	Winkler, Melanie	132
Gupta, Pramod	282	Xiao, David.....	516
Gyorgy, Andras.....	231	Xiao, Lin.....	28
Hazan, Elad.....	144, 257, 314	Xu, Huan.....	490, 503
Honda, Junya	67	Xu, Min	394
Janzing, Dominik.....	464	Zappella, Giovanni	320
Kale, Satyen.....	144, 314	Zeevi, Assaf.....	54
Kanade, Varun	155		
Kivinen, Jyrki	93		
Klivans, Adam	368		
Koenig, Sven	312		
Koltchinskii, Vladimir	420		
Kondor, Risi.....	451		
Kontorovich, Aryeh (Leonid)	433		
Koolen, Wouter M.	93		
Kotlowski, Wojciech	106		
Kötzing, Timo.....	181		
Krause, Andreas.....	333		
Krauthgamer, Robert	433		
Lafferty, John.....	394		
Langford, John.....	316		
Lee, Homin K.	310, 368		
Liu, Han	394		
Loker, David.....	270		
Mannor, Shie	80, 490, 503		
Mansour, Yishay.....	80, 168		
McMahan, H. Brendan	244		

