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 possibily 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 $\mathbb{E}\big[\|Z_t\|^2\big] \le a$ 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

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell(\langle \mathbf{w}_t, \Psi(\mathbf{x}_t)\rangle, y_t) - \min_{\mathbf{w}\in\mathcal{W}} \sum_{t=1}^{T} \ell(\langle \mathbf{w}, \Psi(\mathbf{x}_t)\rangle, y_t)\right] \tag{1}$$

where the random quantities are the predictors $\mathbf{w}_1, \mathbf{w}_2, \ldots$ 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 $\mathbf{w}_{t+1} = P\big(\mathbf{w}_t - \eta_t \nabla_t\big)$, where η_t is a suitably chosen constant which might depend on t; $\nabla_t = \ell'\big(y_t\langle \mathbf{w}_t, \Psi(\mathbf{x}_t)\rangle\big) y_t \Psi(\mathbf{x}_t)$ is the *gradient* of $\ell\big(y_t\langle \mathbf{w}, \Psi(\mathbf{x}_t)\rangle\big)$ 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

$$\mathbb{E}\left[\ell'(y_t\langle \mathbf{w}_t, \Psi(\tilde{\mathbf{x}}_t)\rangle)\Psi(\tilde{\mathbf{x}}_t)\right] \;=\; \mathbb{E}\left[c\tilde{\mathbf{x}}_t\right] = \mathbf{x}_t.$$

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 $\ell'\big(y_t\langle \mathbf{w}_t, \Psi(\mathbf{x}_t)\rangle\big)$ 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 $f : \mathbb{R} \to \mathbb{R}$ 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 $Q'_n(x_1, \ldots, x_n) = \sum_{i=0}^{n} \gamma_{n,i} \prod_{j=1}^{i} x_j$. 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 $x_1, x_2, \ldots, x_N$, and returns $\frac{1}{p_N}\left(Q'_N(x_1, \ldots, x_N) - Q'_{N-1}(x_1, \ldots, x_{N-1})\right)$, where we assume $Q'_0 = 0$. The expected value of this estimator is equal to:

$$\begin{aligned}
&\mathbb{E}_{N, x_1, \ldots, x_N}\left[\frac{1}{p_N}\left(Q'_N(x_1, \ldots, x_N) - Q'_{N-1}(x_1, \ldots, x_{N-1})\right)\right] \\
&= \sum_{n=1}^{\infty} \frac{p_n}{p_n} \mathbb{E}_{x_1, \ldots, x_n}\left[Q'_n(x_1, \ldots, x_n) - Q'_{n-1}(x_1, \ldots, x_{n-1})\right] \\
&= \sum_{n=1}^{\infty} \big(Q_n(\mathbb{E}[X]) - Q_{n-1}(\mathbb{E}[X])\big) = f(\mathbb{E}[X]).
\end{aligned}$$

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 $\gamma_0, \gamma_1, \ldots$. 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 $x_1,x_2,\ldots,x_N$, and return $\theta=\gamma_N\frac{p^{N+1}}{p-1}x_1x_2\cdots x_N$ 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*

$$\mathbb{E}[\theta^2]\le\frac{p}{p-1}f_+^2\left(\sqrt{p\mathbb{E}[X^2]}\right).$$

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

$$\begin{aligned}\mathbb{E}_{N,x_1,\ldots,x_N}\left[\gamma_N^2\frac{p^{2(N+1)}}{(p-1)^2}x_1^2x_2^2\cdots x_N^2\right] &= \sum_{n=0}^{\infty}\frac{(p-1)p^{2(n+1)}}{(p-1)^2p^{n+1}}\gamma_n^2\mathbb{E}_{x_1,\ldots,x_n}\left[x_1^2x_2^2\cdots x_n^2\right]\\ =\frac{p}{p-1}\sum_{n=0}^{\infty}\gamma_n^2p^n\left(\mathbb{E}[X^2]\right)^n &= \frac{p}{p-1}\sum_{n=0}^{\infty}\left(|\gamma_n|\left(\sqrt{p\mathbb{E}[X^2]}\right)^n\right)^2\\ \le\frac{p}{p-1}\left(\sum_{n=0}^{\infty}|\gamma_n|\left(\sqrt{p\mathbb{E}[X^2]}\right)^n\right)^2 &= \frac{p}{p-1}f_+^2\left(\sqrt{p\mathbb{E}[X^2]}\right).\end{aligned}$$

□

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 $\Psi:\mathbb{R}^d\mapsto\mathbb{R}^{d^2}$,

[1]Admittedly, 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 $\Psi(\mathbf{x}) = (x_1x_1, x_1x_2, \ldots, x_dx_d)$, 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 $(\tilde{x}_1\tilde{x}'_1, \tilde{x}_1\tilde{x}'_2, \ldots, \tilde{x}_d\tilde{x}'_d)$ 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 $i = 1, \ldots, T$.
    $\tilde{\Psi}(\mathbf{x}_i)$ for all $i = 1, \ldots, T$
        // $\tilde{\Psi}(\mathbf{x}_i)$ is a data structure which can store a variable number of vectors in $\mathbb{R}^d$
For $t = 1 \ldots T$
    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 $\tilde{\Psi}(\mathbf{x}_t) :=$ Map_Estimate$(A_t, p)$       // Get unbiased estimate of $\Psi(\mathbf{x}_t)$ in the RKHS
    Let $\tilde{g}_t :=$ 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 $\tilde{n}_t := \sum_{i=1}^{t}\sum_{j=1}^{t} \alpha_{t,i}\alpha_{t,j}$Prod$(\tilde{\Psi}(\mathbf{x}_i), \tilde{\Psi}(\mathbf{x}_j))$
            // Compute squared norm, where 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 $i = 1, \ldots, t$
```

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 $\tilde{\Psi}(\mathbf{x}_1), \ldots, \tilde{\Psi}(\mathbf{x}_t)$ and the values of $\alpha_1, \ldots, \alpha_t$ at round t. To apply this hypothesis on a given instance $\mathbf{x}$, we compute

$\sum_{i=1}^{t}\alpha_{t,i}\texttt{Prod}(\tilde{\Psi}(\mathbf{x}_i),\mathbf{x}')$, where $\texttt{Prod}(\tilde{\Psi}(\mathbf{x}_i),\mathbf{x}')$ 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 $\mathbb{E}[\|\tilde{\mathbf{x}}_t\|^2]\le B_{\tilde{\mathbf{x}}}$ for any $\tilde{\mathbf{x}}_t$ returned by the oracle at round t, for all $t=1,\ldots,T$. 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 $u=B_{\mathbf{w}}\left(\frac{p}{p-1}\right)^2Q(pB_{\tilde{\mathbf{x}}})$, then*

$$\mathbb{E}\left[\sum_{t=1}^{T}\ell(y_t\langle\mathbf{w}_t,\Psi(\mathbf{x}_t)\rangle)-\min_{\mathbf{w}\,:\,\|\mathbf{w}\|^2\le B_{\mathbf{w}}}\sum_{t=1}^{T}\ell(y_t\langle\mathbf{w},\Psi(\mathbf{x}_t)\rangle)\right]\ \le\ \ell'_+(\sqrt{(p-1)u})\sqrt{uT}\ .$$

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 $\mathbb{E}[\|\tilde{\mathbf{x}}_t\|^2]\le B_{\tilde{\mathbf{x}}}$ for any $\tilde{\mathbf{x}}_t$ returned by the oracle at round t, for all $t=1,\ldots,T$. 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

$$\mathbb{E}\left[\sum_{t=1}^{T}\ell(y_t\langle\mathbf{w}_t,\Psi(\mathbf{x}_t)\rangle)-\min_{\mathbf{w}\,:\,\|\mathbf{w}\|^2\le B_{\mathbf{w}}}\sum_{t=1}^{T}\ell(y_t\langle\mathbf{w},\Psi(\mathbf{x}_t)\rangle)\right]\ \le\ \ell'_+(\sqrt{(p-1)u})\sqrt{uT}\ .$$

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 $j = 1, \ldots, n$
    Let $\tilde{\Psi}(\mathbf{x}_t)_j :=$ 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 $\textit{Prod}(\tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}'))$ returns $\langle \tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}')\rangle$ for all $\mathbf{x}, \mathbf{x}'$. Then for any given $\mathbf{w}_t = \alpha_{t-1,1}\tilde{\Psi}(\mathbf{x}_1) + \cdots + \alpha_{t-1,t-1}\tilde{\Psi}(\mathbf{x}_{t-1})$ it holds that*

$$\mathbb{E}_t[\tilde{g}_t] = y_t \ell'(y_t\langle \mathbf{w}_t, \Psi(\mathbf{x}_t)\rangle) \qquad \text{and} \qquad \mathbb{E}_t[\tilde{g}_t^2] \leq \frac{p}{p-1}\ell'_+\left(\sqrt{pB_{\mathbf{w}}B_{\tilde{\Psi}(\mathbf{x})}}\right)^2$$

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

$$\mathbb{E}_t\left[\left(\langle \mathbf{w}_t, \tilde{\Psi}(\mathbf{x}_t)\rangle\right)^2\right] \leq \|\mathbf{w}_t\|^2 \, \mathbb{E}_t\left[\|\tilde{\Psi}(\mathbf{x}_t)\|^2\right] \leq B_{\mathbf{w}} B_{\tilde{\Psi}(\mathbf{x})} \ .$$

□

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 $\ell'_+\big(\sqrt{(p-1)u}\big) = 2\sqrt{(p-1)u}$.*

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 $\ell'_+\big(\sqrt{(p-1)u}\big) = e^{\sqrt{(p-1)u}}$.*

Example 3. *Consider a "smoothed" absolute loss function $\ell_\sigma(\langle \mathbf{w}, \Psi(\mathbf{x})\rangle - y)$, defined as an antiderivative of* Erf(sa) *for some $s > 0$ (see proof for exact analytic form). Then we have that $\ell'_+\big(\sqrt{(p-1)u}\big) \leq \frac{1}{2} + \frac{1}{s\sqrt{\pi(p-1)u}}\left(e^{s^2(p-1)u} - 1\right)$.*

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 $\ell'_+\big(\sqrt{(p-1)u}\big) \leq \frac{2}{s\sqrt{\pi(p-1)u}}\left(e^{s^2(p-1)u-1}\right)$.*

For any s, the loss function in the last two examples are convex, and respectively approximate the absolute loss $\big|\langle \mathbf{w}, \Psi(\mathbf{x})\rangle - y\big|$ and the hinge loss $\max\big\{0, 1 - y\langle \mathbf{w}, \Psi(\mathbf{x})\rangle\big\}$ 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.

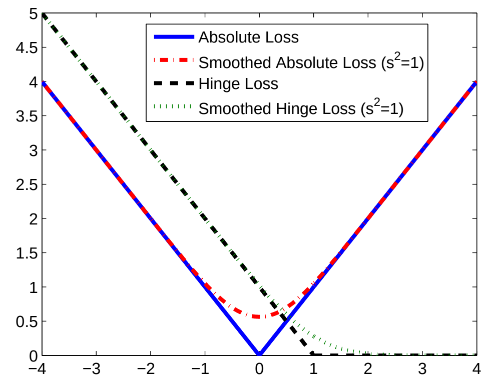

Figure 1: Absolute loss, hinge loss, and smooth approximations

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 $\mathbf{w}_1, \mathbf{w}_2, \ldots$ of predictors output by the algorithm satisfies*

$$\limsup_{T\to\infty} \max_{\mathbf{w}\in\mathcal{W}} \frac{1}{T} \sum_{t=1}^{T} \Big(\ell(\langle \mathbf{w}_t, \mathbf{x}_t \rangle, y_t) - \ell(\langle \mathbf{w}, \mathbf{x}_t \rangle, y_t) \Big) > 0$$

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.

[2] In 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 $\mathbb{E}[\|\tilde{\Psi}(\mathbf{x}_t)\|^2] \le B_{\tilde{\Psi}}$.*
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 $\mathbb{E}[\tilde{g}_t^2] \le B_{\tilde{g}}$.*
4. *For any pair of instances $\mathbf{x}, \mathbf{x}'$, $\texttt{Prod}(\tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}')) = \langle \tilde{\Psi}(\mathbf{x}), \tilde{\Psi}(\mathbf{x}')\rangle$.*

Then if Algorithm 1 is run with $\eta = \sqrt{\frac{B_{\mathbf{w}}}{B_{\tilde{g}} B_{\tilde{\Psi}}}}$, the following inequality holds

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell\big(y_t\langle \mathbf{w}_t, \Psi(\mathbf{x}_t)\rangle\big) - \min_{\mathbf{w}\,:\,\|\mathbf{w}\|^2 \le B_{\mathbf{w}}} \sum_{t=1}^{T} \ell\big(y_t\langle \mathbf{w}, \Psi(\mathbf{x}_t)\rangle\big)\right] \le \sqrt{B_{\mathbf{w}} B_{\tilde{g}} B_{\tilde{\Psi}} T}\ .$$

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 $\beta_n \ge 0$ (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 $k_1, \ldots, k_n \in \{1, \ldots, d\}$, with the entry corresponding to $n, k_1, \ldots, k_n$ 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 $\tilde{\mathbf{x}}^{(1)}, \ldots, \tilde{\mathbf{x}}^{(N)}$, and formally define $\tilde{\Psi}(\mathbf{x})$ as

$$\tilde{\Psi}(\mathbf{x}) \;=\; \sqrt{\beta_n} \frac{p^{n+1}}{p-1} \sum_{k_1=1}^{d} \cdots \sum_{k_n=1}^{d} \tilde{x}_{k_1}^{(1)} \cdots \tilde{x}_{k_n}^{(n)} \mathbf{e}_{n,k_1,\ldots,k_n} \tag{2}$$

where $\mathbf{e}_{n,k_1,\ldots,k_n}$ represents the unit vector in the direction indexed by $n, k_1, \ldots, k_n$ as explained above. Since the oracle queries are i.i.d., the expectation of this expression is

$$\sum_{n=0}^{\infty} \frac{p-1}{p^{n+1}} \sqrt{\beta_n} \frac{p^{n+1}}{p-1} \sum_{k_1=1}^{d} \cdots \sum_{k_n=1}^{d} \mathbb{E}\big[\tilde{x}_{k_1}^{(1)} \cdots \tilde{x}_{k_n}^{(n)}\big] \mathbf{e}_{n,k_1,\ldots,k_n} = \sum_{n=0}^{\infty} \sum_{k_1=1}^{d} \cdots \sum_{k_n=1}^{d} \sqrt{\beta_n} x_{k_1}^{(1)} \cdots x_{k_n}^{(n)} \mathbf{e}_{n,k_1,\ldots,k_n}$$

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),

$$\mathbb{E}\left[\|\tilde{\Psi}(\mathbf{x}_t)\|^2\right] = \mathbb{E}\left[\beta_n \frac{p^{2n+2}}{(p-1)^2} \sum_{k_1\ldots,k_n=1}^{d} \left(\tilde{x}^{(1)}_{t,k_1}\cdots \tilde{x}^{(N)}_{t,k_n}\right)^2\right] = \mathbb{E}\left[\beta_n \frac{p^{2n+2}}{(p-1)^2} \prod_{j=1}^{n} \|\tilde{\mathbf{x}}_t^{(j)}\|^2\right]$$

$$= \sum_{n=0}^{\infty} \frac{p-1}{p^{n+1}} \beta_n \frac{p^{2n+2}}{(p-1)^2} \left(\mathbb{E}[\tilde{\mathbf{x}}_t^2]\right)^n = \frac{p}{p-1} \sum_{n=0}^{\infty} \beta_n \left(p\mathbb{E}[\tilde{\mathbf{x}}_t^2]\right)^n \leq \frac{p}{p-1} \sum_{n=0}^{\infty} \beta_n \left(pB_{\tilde{\mathbf{x}}}\right)^n = \frac{p}{p-1} Q(pB_{\tilde{\mathbf{x}}})$$

where the second-to-last step used the fact that $\beta_n \geq 0$ for all n. □

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 $\tilde{\mathbf{x}}_t^{(1)}, \ldots, \tilde{\mathbf{x}}_t^{(N)}$. 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 $\tilde{\mathbf{x}}_t^{(1)}, \ldots, \tilde{\mathbf{x}}_t^{(N)}$
    Return $\tilde{\mathbf{x}}_t^{(1)}, \ldots, \tilde{\mathbf{x}}_t^{(N)}$ 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 explicitely 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 $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ be the index and vectors comprising $\Psi(\mathbf{x})$
    Let $\mathbf{x}'^{(1)}, \ldots, \mathbf{x}'^{(n')}$ 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

$$\langle \tilde{\Psi}(\mathbf{x}) \tilde{\Psi}(\mathbf{x}')\rangle = \beta_n \frac{p^{2n+2}}{(p-1)^2} \sum_{k_1,\ldots,k_n=1}^{d} \tilde{x}^{(1)}_{k_1} \cdots \tilde{x}^{(n)}_{k_n}\, \tilde{x}'^{(1)}_{k_1} \cdots \tilde{x}'^{(n)}_{k_n}$$

$$= \beta_n \frac{p^{2n+2}}{(p-1)^2} \left(\sum_{k_1=1}^{d} \tilde{x}^{(1)}_{k_1} \tilde{x}'^{(1)}_{k_1}\right) \cdots \left(\sum_{k_N=1}^{d} \tilde{x}^{(n)}_{k_N} \tilde{x}'^{(n)}_{k_N}\right) = \beta_n \frac{p^{2n+2}}{(p-1)^2} \prod_{j=1}^{N} \left(\langle \tilde{\mathbf{x}}^{(j)}, \tilde{\mathbf{x}}'^{(j)}\rangle\right)$$

which is exactly what the algorithm returns, hence the lemma follows. □

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 `Prod`$(\tilde{\Psi}(\mathbf{x}), \mathbf{x}')$

```
Let $n, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ 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 $\left(1 + \frac{1}{p-1}\right)\frac{1}{p-1} = \frac{p}{(p-1)^2}$.

In terms of running time, we note that the expected running time of `Prod` is $O\big(1 + \frac{d}{p-1}\big)$, 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 $O\big(1 + \frac{1}{p-1}\big)$. The expected running time of `Grad_Length_Estimate` is $O\big(1 + \frac{1}{p-1}\big(1 + \frac{1}{p-1}\big) + T\big(1 + \frac{d}{p-1}\big)\big)$, which can be written as $O\big(\frac{p}{(p-1)^2} + T\big(1 + \frac{d}{p-1}\big)\big)$. 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*

$$\operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \mathbb{E}\big[\ell(\langle \mathbf{w}, \mathbf{x}\rangle, 1)\big] \quad \textit{and} \quad \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \ell\big(\langle \mathbf{w}, \mathbb{E}[\mathbf{x}]\rangle, 1\big) \tag{3}$$

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

$$\limsup_{T\to\infty} \max_{\mathbf{w}\in\mathcal{W}} \frac{1}{T} \sum_{t=1}^{T} \Big(\ell(\langle \mathbf{w}_t, \mathbf{x}_t\rangle, y_t) - \ell(\langle \mathbf{w}, \mathbf{x}_t\rangle, y_t)\Big) > 0$$

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 $(1, 0, \ldots, 0)$. Thus, it is enough to show that

$$\operatorname*{argmin}_{w\,:\,|w|^2 \leq B_{\mathbf{w}}} \ell(3w, 1) + \ell(-w, 1) \qquad \text{and} \qquad \operatorname*{argmin}_{w\,:\,|w|^2 \leq B_{\mathbf{w}}} \ell(w, 1) \tag{4}$$

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.