# 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 $\mathrm{OPT}_{disj}$ of the best disjunction with respect to $(\mathcal{D}, c^*)$. Achieving error $O(n \cdot \mathrm{OPT}_{disj}) + \epsilon$ is trivial, and Winnow [13] achieves error $O(r \cdot \mathrm{OPT}_{disj}) + \epsilon$, where $r$ is the number of relevant variables in the best disjunction. In recent work, Peleg [14] shows how to achieve a bound of $\tilde{O}(\sqrt{n} \cdot \mathrm{OPT}_{disj}) + \epsilon$ in polynomial time. In this paper we improve on Peleg's bound, giving a polynomial-time algorithm achieving a bound of

$$O(n^{1/3+\alpha} \cdot \mathrm{OPT}_{disj}) + \epsilon$$

for any constant $\alpha > 0$. The heart of the algorithm is a method for weak-learning when $\mathrm{OPT}_{disj} = O(1/n^{1/3+\alpha})$, 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 $\epsilon$ (ignoring the logarithmic dependence on the confidence term $\delta$). 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 $\mathbf{err}_{\mathcal{D}}(h) = \Pr_{\mathcal{D}}(h(x) \neq c^*(x))$ satisfies $\mathbf{err}_{\mathcal{D}}(h) \leq c \cdot \mathrm{OPT}_{disj} + \epsilon$, where $\mathrm{OPT}_{disj}$ 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 disjunction[1] [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 $\mathrm{OPT}_{disj}$ of the best disjunction is at most $\epsilon$ 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 $\mathrm{OPT}_{disj}$ 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 $\tilde{O}(\sqrt{n} \cdot \mathrm{OPT}_{disj}(S))$.[2]

[1] The 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.

[2] His 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 $O(n^{1/3+\alpha} \cdot \mathrm{OPT}_{disj}) + \epsilon$ 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 $\mathrm{poly}(1/\epsilon, \log(1/\delta), n)$ random examples from $\mathcal{D}$, and outputs a hypothesis $h$, such that with probability $> 1-\delta$,*

$$\mathbf{err}_{\mathcal{D}}(h) \leq O(n^{\frac{1}{3}+\alpha}\mathrm{OPT}_{disj}) + \epsilon$$

*where* $\mathrm{OPT}_{disj} = \min_{f \in \mathrm{DISJUNCTIONS}} \mathbf{err}_{\mathcal{D}}(f)$.

The proof of Theorem 1 is based on finding a weak-learner under the assumption that $\mathsf{OPT} \equiv \mathrm{OPT}_{disj} = O(n^{-(1/3+\alpha)})$. 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 $\mathsf{OPT} < n^{-\frac{1}{3}-\alpha}$, 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 $\mathsf{OPT} = \mathbf{err}_{\mathcal{D}}(c^{opt})$. Call an example $x$ "good" if $c^*(x) = c^{opt}(x)$ and "bad" otherwise. Now, since the only negative examples than can have a relevant $x_i$ set to 1 are the bad negatives, this means that for relevant variables $i$, $\Pr_{x\sim\mathcal{D}}(x_i = 1 | c^*(x) = -1) = O(\mathsf{OPT})$. Therefore, $\Pr_{x\sim\mathcal{D}}(x_i = 1 | c^*(x) = +1) = O(\mathsf{OPT})$ and so $\Pr_{x\sim\mathcal{D}}(x_i = 1) = O(\mathsf{OPT})$ 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 $\omega(\mathsf{OPT})$ 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 $O(\mathsf{OPT})$, this means the average number of variables set to 1 per example is $O(\mathsf{OPT} \cdot n)$. 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 $\mathsf{OPT} = o(1/\sqrt{n})$, 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 $1 - O(\mathsf{OPT})$ 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 $\Omega(1/\mathsf{OPT})$ relevant variables set to 1. But this is not possible since all examples in $S'$ have only $O(\mathsf{OPT} \cdot n)$ variables set to 1, and $1/\mathsf{OPT} \gg \mathsf{OPT} \cdot n$ for $\mathsf{OPT} = o(1/\sqrt{n})$. So, some hypothesis of the form: "if $x \notin S'$ then flip a fair coin, else predict $x_i$" must be a weak-learner.

[3] This 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 $\theta$, 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 $c^* : \mathcal{X} \to \{1, -1\}$, 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 $\mathrm{OPT}_{disj}$ or simply $\mathsf{OPT}$. I.e., $\mathsf{OPT} = \Pr_{x\sim\mathcal{D}}[c^{opt}(x) \neq c^*(x)]$. For the rest of the paper, we will assume that $\mathsf{OPT} = \Omega(\frac{1}{\sqrt{n}})$ (otherwise we can use Peleg's algorithm described in the previous section). We will also assume that we know the value of $\mathsf{OPT}$.[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 $\mathsf{OPT}$. Given this, we may assume that $m^+_{bad} \leq m \cdot \mathsf{OPT}(1 + o(1))$ and $m^-_{bad} \leq m \cdot \mathsf{OPT}(1 + o(1))$.

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 $\mathsf{OPT}$ 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

[4] E.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.

[5] If $\mathsf{OPT}$ is unknown, we can efficiently enumerate over possible guesses for $\mathsf{OPT}$ 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 $\mathsf{OPT} < n^{-\frac{1}{3}-\alpha}$, 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 $\gamma = \Omega(n^{-2})$.*

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 $\tilde{m} \geq m/\mathrm{polylog}(n)$ 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 $\mathsf{OPT}(1 + o(1))$, the fraction of bad *positive* examples may be as large as $1 - 1/\mathrm{polylog}(n)$. Nonetheless, this will still allow for weak learning.

### 3.1 $(B, \alpha, \tilde{m})$-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 $(B, \alpha, \tilde{m})$-*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 $\frac{nB}{m} \cdot m^-_{bad}(1+o(1)) \leq nB \cdot \mathsf{OPT}(1+o(1))$ 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 $n^{1+\alpha}B\mathsf{OPT}$.

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 $1 - O(\mathsf{OPT})$.[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 $m\mathsf{OPT}$ 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 $\frac{\mathsf{OPT}}{n^{-1/3}}$. Furthermore, the ratio $\frac{\tilde{m}}{m}$ will play a role in the definition of $\gamma$, our advantage over a random guess.

An instance satisfying all the above three properties is called a $(B, \alpha, \tilde{m})$-*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 $\theta$, we define the* $(e, \theta)$-restricted domain *to be the set of all examples whose intersection with $e$ is strictly smaller than $\theta$. That is, the set of examples $x$ such that $x \cdot e < \theta$. For any hypothesis $h$, we define the* $(e, \theta)$-restricted hypothesis *to be $h$ over any example*

[6] Indeed, if the original instance was sparse, we would have $\tilde{m} = m(1 - o(1))$.

*that belongs to the $(e, \theta)$ restricted domain, and "I don't know" (flipping a fair coin) over any other example. In particular, we consider the*

- *$(e, \theta)$-restricted $(+1)$-hypothesis – predict $+1$ if the given example intersects $e$ on less than $\theta$ variables.*
- *$(e, \theta)$-restricted $(-1)$-hypothesis – predict $-1$ if the given example intersects $e$ on less than $\theta$ variables.*
- *$(e, \theta)$-restricted $x_i$-hypothesis – predict $+1$ if the given example intersects $e$ on less than $\theta$ variables* and *has $x_i = 1$.*

*We call these $n + 2$ restricted hypotheses the* $(e, \theta)$-restricted base hypotheses.

Our weak-learning algorithm enumerates over all pairs of $(e, \theta)$, where $e$ is a negative example in our training set and $\theta$ is an integer between 1 and $n$. For every such pair, our algorithm checks whether any of the $n + 2$ restricted hypothesis is a $\Omega(\frac{\tilde{m}}{m} \cdot \frac{\mathsf{OPT}}{r})$-weak-learner (see Algorithm 1 below). Our next lemma proves that for $(B, \alpha, \tilde{m})$-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 $\theta$.

**Algorithm 1** A weak learner for sparse instances.

**Input:** A $(B, \alpha, \tilde{m})$ sparse instance.

**Step 1:** For every negative example $e$ in the set and every $\theta \in \{1, 2, \ldots, n\}$

**Step 1a:** Check if any of the $(e, \theta)$-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 $(B, \alpha, \tilde{m})$-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 $\theta$ such that one of the $(e, \theta)$-restricted base hypotheses mentioned in Definition 3 has error at most $1/2 - \gamma$ for $\gamma = \Omega(\frac{\tilde{m}}{m} \cdot \frac{\mathsf{OPT}}{r})$. Since we may assume $\mathsf{OPT} > 1/\sqrt{n}$, 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 $B = O(m\mathsf{OPT})$, because relevant variables have no more than $O(m\mathsf{OPT})$ 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 $r\mathsf{OPT} = \Omega(\tilde{m}/m)$. We now show how to find a weak learner given a $(B, \alpha, \tilde{m})$-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 $\theta_i$ 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 $\theta_i$ 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 $(e_i, \theta_i)$ 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 $(e_i, \theta_i)$-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 $\theta_i$ such ones, the overall contribution on the negative side is $\leq \theta_i \cdot m\mathsf{OPT}(1 + o(1)) = \beta \frac{t_i B'}{m^+} \cdot m\mathsf{OPT}(1 + o(1))$. 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 $2\beta \frac{m}{m^+} \cdot t_i B \cdot \mathsf{OPT}$ (see below, at the specification of the value of $\gamma$). So even if each occurrence of 1 comes from a unique positive example, we still have no more

than $2\beta\frac{m}{m^+}\cdot t_i B\cdot \mathsf{OPT}$ positive examples from the $(e_i,\theta_i)$ 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 $2\beta\frac{m}{m^+}\cdot t_i B\cdot \mathsf{OPT}+m^+/\beta$ 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 $m\mathsf{OPT}$ 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 $2\beta\frac{m}{m^+}\cdot t_i B\cdot\mathsf{OPT}+m^+/\beta$, the total number of edges is at most $2\beta\frac{m}{m^+}B\mathsf{OPT}\sum_i t_i+m^+\cdot m\mathsf{OPT}/\beta$, and therefore some good positive examples must have degree at most $\mathsf{OPT}[\frac{2\beta Bm}{\tilde{m}m^+}\sum_i t_i+\frac{m^+}{\beta}\cdot\frac{m}{\tilde{m}}]$. On the other hand, since we are given a $(B,\alpha,\tilde{m})$-sparse instance, we know that every good positive example intersects *at least* $\frac{B(1-o(1))}{2}$ negative examples, and moreover that $\sum_i t_i\leq n^{1+\alpha}B\mathsf{OPT}$. Putting this together we have:

$$B/2 \quad\leq\quad (1+o(1))\mathsf{OPT}\left[\frac{2\beta B^2 n^{1+\alpha}\mathsf{OPT}m}{\tilde{m}m^+}+\frac{m^+}{\beta}\cdot\frac{m}{\tilde{m}}\right].$$

Setting $\beta=\sqrt{\frac{(m^+)^2}{2B^2n^{1+\alpha}\mathsf{OPT}}}$ to equalize the two terms in the sum above, we derive

$$B\leq\quad 4\sqrt{2}(1+o(1))B\cdot\tfrac{m}{\tilde{m}}\cdot n^{(1+\alpha)/2}\mathsf{OPT}^{3/2}.$$

Thus we have $n^{1+\alpha}\cdot\frac{m^2}{\tilde{m}^2}\cdot\mathsf{OPT}^3\geq\frac{1+o(1)}{32}$. Recall that $\tilde{m}/m\geq n^{-o(\alpha)}$, so we derive a contradiction, as for sufficiently large $n$ it must hold that

$$\mathsf{OPT}\geq\left(\frac{1+o(1)}{32}\right)^{1/3}n^{-\frac{1+\alpha}{3}-o(\alpha)}>n^{-1/3-\alpha}.$$

In order to complete the proof, we need to verify that indeed $\beta>1$. Recall $B=O(m\mathsf{OPT})$ and $m^+\geq\tilde{m}$, so $m^+/m\geq n^{-o(\alpha)}$. Thus $\beta^2=\Omega(\frac{1}{n^{1+\alpha+o(\alpha)}\mathsf{OPT}^3})=\Omega(n^{2\alpha-o(\alpha)})$ by our assumption on $\mathsf{OPT}$.

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 $2\beta\frac{m}{m^+}\cdot t_i B\cdot\mathsf{OPT}$, whereas on the negative side, there can be at most $\beta\frac{m}{m^+}\cdot t_i B\cdot\mathsf{OPT}(1+o(1))$ 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 $\frac{\beta\cdot t_i Bm\cdot\mathsf{OPT}(1-o(1))}{m^+\min(r,t_i)}>B\cdot\mathsf{OPT}(1-o(1))$ since $\beta>1$. Finally, using the fact that $B=\Omega(\tilde{m}/r)$ we get a gap of $\Omega(\frac{\tilde{m}\mathsf{OPT}}{r})$ or equivalently an advantage of $\gamma=\Omega(\frac{\mathsf{OPT}}{r}\cdot\frac{\tilde{m}}{m})$. This advantage is trivially $\Omega(n^{-2(1+o(\alpha))})$, or, using the assumption $\mathsf{OPT}>1/\sqrt{n}$ (for otherwise, we can apply Peleg's algorithm [14]), we get $\gamma=\Omega(n^{-\frac{3}{2}(1+o(\alpha))})$. ■

### 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 $(B,\alpha,\tilde{m})$-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 $\mathrm{polylog}(n)$-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 $(B,\alpha,\tilde{m})$-sparse. Formally, we prove this next lemma.

**Lemma 5** *There exists a $\mathrm{poly}((\log n)^{O(1/\alpha)},n,m)$-time algorithm, that gets as an input $2m$ labeled examples in $\{0,1\}^n$, and output a list of subsets, each containing $m/\mathrm{polylog}(n)$ examples, s.t. either some subset has a trivial weak-learner, or some subset is $(B,\alpha,m/\mathrm{polylog}(n))$-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 $(B,\alpha,\tilde{m})$-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 $m^-_{bad}\leq m\cdot\mathsf{OPT}(1+o(1))$ 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 respextively. 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 $\geq (n')^{1+\alpha}(B')\mathsf{OPT}$. 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 $r\mathsf{OPT} \ll 1$, we have an immediate weak-learner, so it must hold that in the $i$-th step, we still retain at least $n_i = 1/\mathsf{OPT}$ 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 $(n_i)^{1+\alpha}(B')\mathsf{OPT}$ ones before we threw out examples. Once we remove dense examples, they contain no more than $\frac{2(n_i)(B')}{m_i} \cdot m\mathsf{OPT}$ 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 $1/\mathsf{OPT} > n^{1/3}$, 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 $\mathrm{poly}(n, m)$ time for the bucketing, removal and checking for weak-learner. We conclude that the run-time of this algorithm is $\mathrm{poly}((\log n)^{1/\alpha}, n, m)$. ■

**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 $2m\mathsf{OPT}$ 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 $(\eta, \gamma)$-weak learners. In our context an $(\eta, \gamma)$-weak learner is an algorithm which with respect to to any distribution $\mathcal{D}$, with high probability, produces a hypothesis of error $\leq \frac{1}{2} - \gamma$, whenever $\mathrm{OPT}_{disj} \leq \frac{1}{2} - \eta$.

**Theorem 6 (Feldman [4], Theorem 3.5)** *There exists an algorithm* ABoostDI *that, given a $(\eta, \gamma)$-weak learner, for every distribution $\mathcal{D}$ and $\epsilon > 0$, produces, with high probability, a hypothesis $h$ such that $err_D(h) \leq \frac{\mathrm{OPT}_{disj}}{1-2\eta} + \epsilon$. Furthermore, the running time of the algorithm is $T \cdot poly(\frac{1}{\gamma}, \frac{1}{\epsilon})$, 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 $\mathbf{err}_{\mathcal{D}}(h) \leq 2n^{1/3+\alpha}\mathsf{OPT} + \epsilon$. 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 $\mathsf{OPT} < n^{-(1/3+\alpha)}$, achieving an error rate of $O(n^{1/3+\alpha} \cdot \mathsf{OPT}) + \epsilon$. The natural open question is whether one can improve this bound. For example, can one achieve weak agnostic learning for $\mathsf{OPT} = n^{-1/4}$? 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 $\mathsf{OPT} = n^{-0.99}$. 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 $t/\mathsf{OPT}$ 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 $r/2 + t/\mathsf{OPT}$ 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 $\theta$ 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.