On Agnostic Boosting and Parity Learning

Adam Tauman Kalai[*]
College of Computing
Georgia Inst. of Technology
Atlanta, GA, USA
atk@cc.gatech.edu

Yishay Mansour[†]
Google Inc.
New York, NY, USA
and
School of Computer Science
Tel Aviv University
Tel Aviv, Israel
mansour@tau.ac.il

Elad Verbin[‡]
Institute for Theoretical
Computer Science
Tsinghua University
Beijing, China
eladv@tsinghua.edu.cn

ABSTRACT

The motivating problem is agnostically learning parity functions, i.e., parity with arbitrary or adversarial noise. Specifically, given random labeled examples from an *arbitrary* distribution, we would like to produce an hypothesis whose accuracy nearly matches the accuracy of the best parity function. Our algorithm runs in time $2^{O(n/\log n)}$, which matches the best known for the easier cases of learning parities with random classification noise (Blum *et al*, 2003) and for agnostically learning parities over the uniform distribution on inputs (Feldman *et al*, 2006).

Our approach is as follows. We give an *agnostic boosting* theorem that is capable of nearly achieving optimal accuracy, improving upon earlier studies (starting with Ben David *et al*, 2001). To achieve this, we circumvent previous lower bounds by altering the boosting model. We then show that the (random noise) parity learning algorithm of Blum *et al* (2000) fits our new model of agnostic weak learner. Our agnostic boosting framework is completely general and may be applied to other agnostic learning problems. Hence, it also sheds light on the actual difficulty of agnostic learning by showing that full agnostic boosting is indeed possible.

[*]Research partly performed while visiting the Weizmann Institute, and supported in part by NSF SES-0734780.

[†]This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, by a grant no. 1079/04 from the Israel Science Foundation, by a grant from United States-Israel Binational Science Foundation (BSF), and an IBM faculty award. This publication reflects the authors' views only.

[‡]Some of this research was done while at Tel Aviv University, Tel Aviv, Israel. This work was supported in part by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900,2007CB807901.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*

General Terms

Algorithms, Theory

Keywords

agnostic learning, agnostic boosting, learning parity with noise, sub-exponential algorithms

1. INTRODUCTION

The goal of binary classification is to learn, from labeled examples drawn from a distribution, to predict labels of future examples drawn from the same distribution. In normal PAC learning, one assumes that the labels are determined by a function f that belongs to some restricted class C, such as the class of (small) DNF expressions on the input bits. In *agnostic learning* [16], on the other hand, one avoids making any assumptions relating to the process generating labels for examples. Thus, we allow any or no relationship between labels and examples. Clearly some relaxation of the goal is required, since one can not hope to learn pure random noise. In agnostic learning, the goal of the learner is to return an hypothesis that nearly matches the error of the best classifier from some given class C. Thus, we can see that PAC learning is the special case where the best classifier has zero error, though in most applications of interest the best classifier has non-zero error.

There are two main points to this paper. The first is an agnostic boosting theorem. The boosting algorithm itself is a small modification to boosting by branching programs [14, 12]. It receives a weak learner, which only needs to guarantee an accuracy slightly better than $1/2$. The booster outputs an hypothesis whose error nearly matches the error of the best classifier in the class. The key insight is a novel understanding of what an weak agnostic learner should do, which is what enables us to match the error of the best classifier. We remark that our boosting algorithm makes no assumption regarding the underlying distribution and works for an *arbitrary* distribution.

To demonstrate the power of our agnostic boosting model, we show how one can build an agnostic weak learner for the class of parity functions which works for *arbitrary* distributions. The class of parity functions is notoriously hard to learn even with random classification noise and the uniform distribution, let alone agnostic learning with arbitrary distribution. Not surprisingly, our weak learner for parity does not run in polynomial time but rather $2^{O(n/\log n)}$, which is still faster than the trivial $2^{O(n)}$ bound.

Combining our two results: the agnostic boosting algorithm and the weak agnostic learner for parity function, we get an $2^{O(n/\log n)}$ time algorithm for computing an hypothesis whose error almost matches that of the best parity function. This learning result, although not impressive in the running time, works in a highly noisy environment and handles an arbitrary distribution.

1.1 Learning Models

The *agnostic* model of binary classification [16] is the most general (and adversarial) noise-tolerant extension of PAC learning, and it is also the most computationally demanding. In this model, there is an *arbitrary* distribution D over binary *labeled examples* $(x, y) \in X \times \{0,1\}$. There is also a class C of *concepts* $c : X \to \{0,1\}$. Based on random labeled examples from D, the goal is to output a predictor $h : X \to \{0,1\}$ with low *error* on future examples,

$$\underset{D}{\mathrm{err}}(h) = \mathrm{P}_{(x,y)\sim D}[h(x) \neq y].$$

An agnostic learner's error is required to be near that of the best predictor in C. In particular, it should not be much larger than

$$\underset{D}{\mathrm{opt}}(C) = \min_{c\in C} \underset{D}{\mathrm{err}}(c).$$

In Valiant's original PAC model [22], D was *noiseless*, i.e., $\mathrm{opt}_D(C) = 0$, and PAC learners guarantee $\mathrm{err}_D(h) \leq \epsilon$. In agnostic learning, since D is arbitrary, the goal is to find h with $\mathrm{err}_D(h) \leq \mathrm{opt}_D(C) + \epsilon$. "(The polynomial-time algorithm for PAC-learning parity, based on Gaussian elimination, is the one striking exception to this rule of thumb, since it cannot be made to work in the Statistical Query model [3].)

Classification noise is a model lying in between PAC and agnostic learning. Here, it is assumed that there is a *true* function $f \in C$, and that labels y agree with $f(x)$ with probability $1 - \eta > 1/2$, independently for each x. Most of the efficient PAC learning algorithms can be translated from the noiseless to the classification noise settings using Kearns' Statistical Query model [13].

Classification noise is a special case of agnostic learning ($\mathrm{opt} = \eta$). "Agnostic learning" can be viewed as learning with arbitrary or even adversarial noise. However, Kearns, Schapire, and Sellie [16] chose the term "agnostic learning" because "noise" suggests an assumed connection between the concept class C and the real-world distribution D.

Surprisingly, the three models of learnability are similar in terms of number of examples required, disregarding computation. However, many computationally efficient PAC (with classification noise) learning algorithms are known, while the majority of agnostic learning results have been negative. Even for trivial classes of functions, such as disjunctions over $\{0,1\}^n$, efficient agnostic learning remains a frustrating hard open problem. (For example, agnostic learning of disjunctions would imply PAC learning of DNF [16].) Recently, it was shown that the popular class of halfspaces is agnostically learnable, under mild distributional assumptions [11] (and an exponential dependence on $1/\epsilon$).

In this paper, we give two further positive results for agnostic learning. The first is a polynomial-time general agnostic boosting algorithm. The second is its application to the problem of agnostic parity. Next we will discuss each of the contributions in more detail.

1.2 Agnostic boosting

Boosting is a central tool for the design of efficient PAC-learning algorithms, which has also had a great impact in practice. In the PAC setting, a *weak learner* is an algorithm that guarantees error $\leq 1/2 - \gamma$, for some γ bounded away from 0. It was shown [21] that such a learner could be used repeatedly to achieve error below any $\epsilon > 0$. In the agnostic setting, it is impossible to guarantee error $< 1/2$ (e.g., $y \in \{0,1\}$ may be uniformly random and independent from x).

Ben-David, *et al* [2] were the first to define the agnostic boosting problem. They defined a weak agnostic learner as one that achieves error $\leq \mathrm{opt}_D(C) + \beta$ for some $\beta < 1/2$ and showed that such a weak agnostic learner could be boosted to a certain (non-optimal) amount. In subsequent work, Gavinsky [6] showed that such a learner can be boosted to achieve error $\leq \frac{\mathrm{opt}}{1/2-\beta} + \epsilon$ and gave a lower-bound off by a factor of 2.

We suggest the following new definition of weak learner in the agnostic setting, for $0 < \gamma \leq \alpha \leq 1/2$.

Definition 1. (α, γ)-weak agnostic learner (idealized version). *The learner takes labeled examples from arbitrary distribution D on $X \times \{0,1\}$, and outputs $h : X \to \{0,1\}$ such that,*

$$\left(\underset{D}{\mathrm{opt}}(C) \leq \frac{1}{2} - \alpha\right) \implies \left(\underset{D}{\mathrm{err}}(h) \leq \frac{1}{2} - \gamma\right).$$

Note that a good weak learner has α as small as possible and γ as large as possible. Note also that the above is idealized – actual guarantees can only hold with high probability. This definition differs from the PAC definition, in that the algorithm is only required to succeed when there is some concept in C with error $\leq 1/2 - \alpha$. For $\alpha = 1/2$, this matches the standard definition of γ-weak learner in the noiseless PAC setting.

Also note that one cannot expect a (α, γ)-weak agnostic learner to help in guaranteeing error $< \mathrm{opt} + \alpha$ (just consider an input distribution where $\mathrm{opt} > 1/2 - \alpha$, so the agnostic weak learner guarantee is vacuous). Given any (α, γ) weak agnostic learner, our algorithm produces an hypothesis with error arbitrarily close to $\mathrm{opt} + \alpha$.

Theorem 1 (Informal Statement). *Given any (α, γ)-weak agnostic learner and any $\epsilon > 0$, with high probability the boosting algorithm outputs h with*

$$\underset{D}{\mathrm{err}}(h) \leq \mathrm{opt} + \alpha + \epsilon.$$

The algorithm makes $\mathrm{poly}(\gamma^{-1}, \epsilon^{-1})$ *calls to the weak learner.*

The algorithm we use is essentially a boosting algorithm due to Mansour and McAllester [18]. This algorithm has been used in the context of boosting with classification noise [12].

In the examples of weak learners we give in this paper, one can pick the parameter α to be arbitrarily small. However, one can envision applications in which this parameter would be essential to achieve the weak learning.

We next illustrate through the class of unions of n intervals on $X = \mathrm{R}$. While this class is already known to be learnable [16], it serves the purpose of demonstrating how our agnostic boosting theorem can be used to get fully polynomial-time learners. We then move on to the more challenging application of agnostic learning of parity functions.

1.3 Agnostic learning of parity functions

Learning parity with noise is a longstanding open problem in computational learning theory, which has attracted attention in recent years [4, 5, 20, 9, 10, 19, 17, 7], for several reasons. Firstly, an efficient algorithm, if one exists, can be used [5] to solve outstanding open problems in learning theory, such as the problem of learning DNF, and would have implications for decoding random linear codes, as well. The common belief, however, is that parity with noise is hard, even when D is the uniform distribution [13]. Thus, the goal is to understand *how hard it is.* Parity with noise seems to be a relatively easy problem compared to NP-Hard problems, similarly to *factoring*, *graph isomorphism*, or various lattice problems. Parity with noise also seems to be a difficult-on-average problem, and thus is useful as an hypothesis for achieving conditional hardness-on-average results and as a cryptographic primitive [3, 20, 9, 10]. In either case, parity with noise is a beautiful mathematical problem whose roots are found in Gaussian elimination, and which captures the inherent computational difficulty of dealing with noisy data.

It is important to note that the difficulty of parity with noise is complexity-theoretic, rather than information-theoretic. Indeed, given $m = O(n)$ samples, selecting the parity with the minimal error rate on the sample would give a good approximation to the best parity function.

Let $X = \{0,1\}^n$. A *parity function* is a function $c(x) = \sum_{i=1}^n x_i z_i \pmod 2$ for some $z \in \{0,1\}^n$. Let P_n be the class of parity functions on n bits. Learning P_n seems to be quite challenging even in the case of classification noise. The current fastest algorithm, due to Blum, *et al* [4] runs in time $2^{O(n/\log n)}$ for noise rate $\eta = \frac{1}{2} - 2^{-n^{1-\delta}}$ (any constant $\delta > 0$).[1] Recently Feldman, *et al* [5], showed that any algorithm for learning parity with random classification noise w.r.t. the uniform distribution on $\{0,1\}^n$ can be used to agnostically learn parity (among other things), with a polynomial blowup in the number of examples and in running time, *assuming that the distribution D is uniform over* $x \in \{0,1\}^n$ (and the labels selected in a potentially adversarial way). This implies one can learn agnostic parity in time $2^{O(n/\log n)}$ for the uniform distribution over $x \in \{0,1\}^n$. Our result regarding parity matches this runtime in the general distribution case, and uses a different approach.

THEOREM 2. *There is an algorithm such that, for any $n \geq 1$ and distribution D over $(x, y) \in \{0,1\}^n \times \{0,1\}$, with probability ≥ 0.99, given as input $m = 2^{O(n/\log n)}$ independent samples from D, outputs a circuit computing $h : \{0,1\}^n \to \{0,1\}$ such that,*

$$\operatorname*{err}_D(h) \leq \operatorname*{opt}_D + 2^{-n^{0.99}}$$

The runtime and number of samples used by the algorithm is $2^{O(n/\log n)}$.

[1]These can be found by appropriate parameter settings in Theorem 2 of [4], as observed by Lyubashevsky [17].

Note that the constants 0.99 can be replaced by arbitrary constants less than 1. The circuit produced by the algorithm is of size $2^{O(n/\log n)}$. Our approach is to show that an appropriate modification of the Blum *et al*'s algorithm [4] gives a very weak agnostic learner, i.e., it has a small degree of tolerance to agnostic noise. We note that our algorithm is not a *proper* learner – its output is not a parity function. In contrast the prior results [4, 5] are proper.

2. FORMAL DEFINITIONS

We will write opt and err when the distribution and concept class are understood from context. Formally, there is a domain X and a *learner* is an algorithm that takes as input any number of labeled examples $\in X \times \{0,1\}$, an accuracy parameter ϵ and a confidence parameter δ and outputs a *circuit* computing a function $h : X \to \{0,1\}$.[2]

DEFINITION 2. *Learner A* agnostically learns *concept class C of $c : X \to \{0,1\}$, if there is a polynomial p such that, for every distribution D on $X \times \{0,1\}$ and every $\epsilon, \delta > 0$, given $m = p(1/\epsilon, \log 1/\delta)$ examples, it outputs a circuit computing $h : X \to \{0,1\}$ such that, with probability $\geq 1 - \delta$ over the m examples,*

$$\operatorname*{err}_D(h) \leq \operatorname*{opt}_D(C) + \epsilon.$$

The runtime of the algorithm should be polynomial m, $1/\epsilon$ and $\log(1/\delta)$.

We now give a precise definition of weak agnostic learner.

DEFINITION 3. (m, α, γ)-WEAK AGNOSTIC LEARNER. *For $m \geq 1$, $0 < \gamma \leq \alpha < 1/2$, an (m, α, γ)-weak agnostic learner is a learner that, when given m labeled examples $Z_m = (x_1, y_1), \ldots, (x_m, y_m) \in X \times \{0,1\}$ drawn from arbitrary* distribution *D, outputs a circuit computing $h : X \to \{0,1\}$ such that,*

$$\left(\operatorname*{opt}_D(C) \leq \frac{1}{2} - \alpha\right) \implies \left(\Pr_{Z_m \sim D^m}\left[\operatorname*{err}_D(h) \leq \frac{1}{2} - \gamma\right] \geq 1/2\right)$$

The difference between this definition and the idealized weak agnostic learner is that we have specified the number of examples required and a success rate.

The following is our main theorem regarding our agnostic boosting.

THEOREM 3. *There is an agnostic boosting algorithm, that given any $0 < \gamma < \alpha \leq 1/2$, any (m, α, γ)-weak agnostic learner and $\delta, \epsilon > 0$, with probability $\geq 1 - \delta$ outputs a circuit computing $h : X \to \{0,1\}$ such that*

$$\operatorname*{err}_D(h) \leq \operatorname{opt} + \alpha + \epsilon.$$

The number of samples required is $\operatorname{poly}(m, 1/\gamma, 1/\epsilon, \log(1/\delta))$. *The algorithm makes* $\operatorname{poly}(1/\gamma, 1/\epsilon, \log(1/\delta))$ *calls to the weak learner, and uses additional* $\operatorname{poly}(1/\gamma, 1/\epsilon, \log(1/\delta))$ *runtime.*

The computational bounds of our boosting theorem are stated in terms of the runtime of the weak agnostic learner, which we do not bound *a priori*. This gives us the most

[2]For the purposes of this paper, we consider standard binary circuits with unbounded fan-in. However, our boosting theorem could easily be adapted to real-valued concepts/circuits as well. What is important is that h can be evaluated quickly.

flexibility in its application. In many cases of interest, there is a sequence of concepts classes C_n of $c : X_n \to \{0, 1\}$, for $n = 1, 2, \ldots$. In that case, the algorithm must apply to each n and the required number of training examples m is permitted to be a polynomial in $1/\epsilon$ and n as well. Parity functions do fit into the asymptotic C_n setting. However, our algorithm for parity with noise requires a number of examples that is super-polynomial in n.

It will be more convenient to consider the following model of a learner, which takes some number of labeled examples as input and a single unlabeled example, and predicts the label of that example. Such a learner is similar to an online learner (where an arbitrarily long sequence of problems $m = 1, 2, 3, \ldots$ is given).

Definition 4. (m, α, γ)-weak agnostic guesser. *For $m \geq 1$, $0 < \gamma \leq \alpha < 1/2$, an (m, α, γ)-expected weak agnostic guesser is an algorithm with the following guarantee. Let $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m), (x, y) \in X \times \{0, 1\}$ be drawn independently from arbitrary* distribution D. *The input to the algorithm is $(x_1, y_1), \ldots (x_m, y_m)$,* and x; *the output is a prediction $\hat{y} \in \{0, 1\}$ such that*

$$\left(\underset{D}{\text{opt}}(C) \leq \frac{1}{2} - \alpha\right) \implies \left(\Pr\left[\hat{y} \neq y\right] \leq \frac{1}{2} - \gamma\right),$$

where the above probability over $(x_1, y_1), \ldots, (x_m, y_m), (x, y)$ drawn independently according to D.

It is not hard to see that an expected weak agnostic guesser can be converted to a weak agnostic learner with a polynomial blowup in examples, and a slightly worse guarantee. Given any labeled data set, $Z = (x_1, y_1), \ldots, (x_m, y_m)$, the idea is that we can construct a circuit $Q_Z : X \to \{0, 1\}$ such that $Q_Z(x)$ computes what the weak agnostic guesser would predict on the data set and the unlabeled example x.

Lemma 4. *Let L be an (m, α, γ)-weak agnostic guesser. Then the following is a $\left(\frac{m}{\gamma} + \frac{32}{\gamma^2} \log \frac{20}{\gamma}, \alpha, \gamma/4\right)$-weak agnostic learner. Take $m' = \gamma^{-1}$ independent training sets $Z_1, \ldots, Z_{m'}$ and let circuits $Q_1, \ldots, Q_{m'}$ be such that $Q_i(x)$ computes $L(Z_i, x)$. Using $\frac{32}{\gamma^2} \log \frac{20}{\gamma}$ additional labeled examples, test each Q_i and output the one with minimal empirical error.*

The proof uses standard techniques and is given in the appendix.

2.1 Branching programs

For our purposes, a *branching program* is a rooted, directed acyclic graph in which each leaf ℓ is labeled with a bit b_ℓ and each internal node v has outdegree 2 and is labeled with a Boolean function $h_v : X \to \{0, 1\}$. The branching program computes a value, for each $x \in X$, starting at the root. At any node v (starting at the root) one moves to the child determined by $h_v(x)$, until one reaches a leaf ℓ and the value is b_ℓ. Branching programs were introduced [18] into boosting as a generalization of decision tree learning: while decision trees are constructed by splitting nodes, for branching programs nodes can be merged as well.

For $\ell \subseteq X$ we write $D|_\ell$ to denote D conditioned on $x \in \ell$, i.e. $D|_\ell(S) = \Pr_D[x \in S \mid x \in \ell]$. We write p_ℓ to denote $\Pr_D[y = 1 | x \in \ell]$ and p to denote $\Pr_D[y = 1]$.

Definition 5 ([15]). *Let the* uncertainty *of a distribution D be $U(D) = 2\sqrt{p(1-p)}$. Let $\mathcal{L}$ be a partition of X into disjoint subsets (such that $X = \bigcup_{\ell \in \mathcal{L}} \ell$). The uncertainty of $\mathcal{L}$ under D is $U(\mathcal{L}, D) = \sum_{\ell \in \mathcal{L}} w_\ell u_\ell$, where $u_\ell = U(D|_\ell) = 2\sqrt{p_\ell(1-p_\ell)}$ is the uncertainty of the conditional distribution $D|_\ell$ and $w_\ell = \Pr_D[x \in \ell]$ is referred to as the* weight *of ℓ.*

Definition 6. *The* balanced distribution *of a distribution D, denoted by $\widehat{D}$, is defined as $\widehat{D}(S) = \frac{1}{2}\Pr_D[x \in S \mid y = 1] + \frac{1}{2}\Pr_D[x \in S \mid y = 0]$, i.e., $\widehat{D}$ gives an equal weight to both labels.*

Given access to samples from D, it is easy to simulate random samples from $\widehat{D}$; this is done by flipping a coin at random to decide whether to choose a positive or negative example, and then wait until one receives such an example.[3]

For $\ell_0 \cap \ell_1 = \emptyset$, $\ell = \ell_0 \cup \ell_1$, we write

$$\Delta(\ell_0, \ell_1) = w_\ell u_\ell - w_{\ell_0} u_{\ell_0} - w_{\ell_1} u_{\ell_1}.$$

This *change* is the nonnegative increase in uncertainty of a partition if we were to merge ℓ_0 and ℓ_1, (equivalently, the decrease in uncertainty if we were to split ℓ into ℓ_0, ℓ_1). Kearns and Mansour (Lemma 1) [15] show that,

$$\Delta(\ell_0, \ell_1) = 2\left(\frac{1}{2} - \mathrm{P}_{(x,y)\sim\widehat{D}|_\ell}[x \notin \ell_y]\right)^2 w_\ell u_\ell. \qquad (1)$$

In other words, if $h(x)$ is the classifier that is 1 if $x \in \ell_1$ and 0 if $x \in \ell_0$, then the change in uncertainty is $2(1/2 - \text{err}_{\widehat{D}|_\ell}(h))^2 w_\ell u_\ell$. (This is also Lemma 1 from [12].)

Given any partition $\mathcal{L}$ of X, there is a natural corresponding predictor $\mathcal{B}(\mathcal{L})$: on each set $\ell \in \mathcal{L}$, $\mathcal{B}(\mathcal{L})$ predicts 1 iff $p_\ell > \frac{1}{2}$. The error of $\mathcal{B}(\mathcal{L})$ under D is $\text{err}_D(\mathcal{B}(\mathcal{L})) = \sum_{\ell \in \mathcal{L}} w_\ell \min(p_\ell, 1 - p_\ell)$. Since for $x \in [0, 1]$, $\min(x, 1 - x) \leq \sqrt{x(1-x)}$, the error of $\mathcal{B}(\mathcal{L})$ is at most $\frac{1}{2}U(\mathcal{L}, D)$. Thus, the uncertainty of a partition gives an upper bound on the error of the corresponding predictor, i.e., $\text{err}_D(\mathcal{B}(\mathcal{L})) \leq \frac{1}{2}U(\mathcal{L}, D)$.

3. EXAMPLE: AGNOSTIC LEARNING OF UNIONS OF INTERVALS

We first illustrate that our notion of agnostic boosting is natural on the class of unions of n intervals. Let $X = \mathrm{R}$, and let C_n be the class of functions $f(x) = \mathrm{I}(x \in \bigcup_{i=1}^n I_i)$, where each I_i is an interval (any kind) on the real line.

Claim 5. *The algorithm that, given any $m \geq 1$ labeled examples $(x_i, y_i) \in \mathrm{R} \times \{0, 1\}$, outputs the single interval of minimal empirical error is a (m, α, γ)-weak agnostic learner of C_n for any $\alpha > 0$ and $\gamma = \frac{\alpha}{2n} - c\sqrt{\frac{\log m}{m}}$, where c is a constant.*

The proof is similar to the argument used to show that agnostic learnability of disjunctions implies PAC-learnability of DNF [16].

Proof. Take $f(x) = \mathrm{I}(x \in \bigcup_{i=1}^n I_i)$ that has error opt $\leq \frac{1}{2} - \alpha$. Let ξ be the error of the constant 0 predictor and ξ_i be the error of the function $h_i(x) = \mathrm{I}(x \in I_i)$. Then it

[3]This may take a great deal of time if p is very close to 0 or 1, but as we will see these situations do not pose a problem for us since in such a case we are very close to the all-zeros function or the all-ones function, so we will abort the simulation after some bounded number of draws (see also [8]).

is easy to see that (WLOG assuming the intervals are disjoint), $\sum_{i=1}^{n} \xi_i = \text{opt} + (n-1)\xi$. The reason is that each true error of f is counted once and then each positive example is counted an additional $n-1$ times. Now, if $\xi > \frac{1}{2} + \alpha/(2n)$, then the error of the constant 1 predictor (i.e., the interval $(-\infty, \infty)$) has error $\le \frac{1}{2} - \alpha/(2n)$. Otherwise,

$$\frac{1}{n}\sum_{i=1}^{n} \xi_i = \frac{1}{n}\text{opt} + \frac{n-1}{n}\xi \le \frac{1}{2} - \frac{\alpha}{n} + \frac{\alpha}{2n} = \frac{1}{2} - \frac{\alpha}{2n}.$$

Since one of the ξ_i's must be no larger than their average, there is some interval with error at most $\frac{1}{2} - \frac{\alpha}{2n}$.

By VC theory, with probability at least 1/2, for every interval the difference between the empirical and true error rates is at most $c' \frac{\sqrt{\log m}}{m}$, for some constant $c' > 0$. Therefore, with probability at least 1/2, the interval that minimizes the empirical error has trued error at most $1/2 - \frac{\alpha}{2n} + 2c' \frac{\sqrt{\log m}}{m}$. □

The above, combined with our boosting theorem, implies that unions of n intervals are agnostically learnable in time poly$(n, 1/\epsilon)$. While this class of functions is already known to be agnostically learnable [16] using dynamic programming, the above illustrates how boosting can give very simple solutions and fully polynomial-time algorithms.

4. AGNOSTIC BOOSTING ALGORITHM

In this section, we define and analyze an Agnostic Branching Program Boosting Algorithm (ABPBA), which is a small modification the algorithm of Mansour and McAllester [18] (that builds on ideas from Kearns and Mansour [15]). Our presentation follows closely that of Kalai and Servedio [12], parts verbatim. The main difference between prior versions and ours is that we have simplified the algorithm into alternately splitting and merging pairs of nodes, while Kalai and Servedio had a merging phases in which multiple merges might be performed. The analysis, of course, has to be altered to handle agnostic learning.

4.1 The ABPBA Boosting Algorithm

Following [15, 18], we consider an *idealized* model where exact probabilities can be computed. This simplification is an approximation to reality in the sense that the above probabilities can be estimated to arbitrary precision, and by repetition, such an hypothesis can be achieved with arbitrarily high probability.

The ABPBA algorithm iteratively constructs a branching program in which each internal node v is labeled with an hypothesis h_v generated by the weak learner at some invocation. In such a branching program, any instance $x \in X$ determines a unique directed path from the root to a leaf; at each internal node v the outgoing edge taken depends on the value $h_v(x)$. Thus, the set $\mathcal{L}$ of leaves ℓ corresponds to a partition of X, and for each leaf ℓ we have probabilities $w_\ell = \Pr[x \text{ reaches } \ell]$ and $p_\ell = \Pr_{x \in D}[y = 1 | x \text{ reaches } \ell]$. As described above, each leaf ℓ is labeled 1 if $p_\ell > \frac{1}{2}$ and is labeled 0 otherwise; thus a branching program naturally corresponds to the classifier $\mathcal{B}(\mathcal{L})$.

The ABPBA algorithm is given in Fig. 1. The branching program initially consists of a single leaf. The algorithm repeatedly performs two basic operations:

- **Split a leaf (steps 2a-2b):** The chosen leaf λ becomes an internal node which has two new leaves as its children. The label of this new internal node is an hypothesis generated by the weak learning algorithm when run with the oracle $EX(\widehat{D|_\lambda})$ (recall that this distribution is obtained by first conditioning on whether $x \in \lambda$ (or $x \notin \lambda$) and then balancing that conditional distribution).
- **Merge two leaves (steps 2c-2d):** The two leaves ℓ_a and ℓ_b chosen for the merge are replaced by a single leaf $\ell_{a,b}$. All edges into ℓ_a and ℓ_b are redirected into $\ell_{a,b}$.

Intuitively, splitting a leaf should increase the accuracy of our classifier. In the ABPBA algorithm, the leaf to be split is chosen so as to maximally decrease the overall uncertainty of the partition corresponding to the branching program. Conversely, merging two leaves should decrease the accuracy of our classifier. However, we must do merges in order to ensure that the branching program does not get too large; Kearns and Mansour [15] have shown that without merges (i.e., building a tree) the size of the resulting decision tree may be exponentially large. The leaves to be merged are chosen so as to minimally increase the overall uncertainty of the partition. The condition in **2d** ensures that we only perform a merge whose uncertainty increase is substantially less than the uncertainty decrease of the split, and thus we make progress. The final output hypothesis of the ABPBA booster is the final branching program.

4.2 Agnostic analysis of the ABPBA algorithm

As we mentioned, we assume in this section that all probabilities are computed exactly by the ABPBA algorithm. We also assume that the weak learning algorithm successfully finds a $(\frac{1}{2} - \gamma)$-accurate hypothesis at each invocation, i.e., we ignore the constant probability of failure. (This failure probability can be handled with standard techniques and would have a negligible influence on the result.) First, we observe that the following holds:

LEMMA 6. *For any distribution D over $X \times \{0,1\}$, and family C of $c : X \to \{0,1\}$,*

$$\frac{1}{2} - \underset{\hat{D}}{\text{opt}}(C) \ge \min\{\text{P}_D[y=0], \text{P}_D[y=1]\} - \underset{D}{\text{opt}}(C).$$

All proofs in this section are deferred to Appendix D. We then show that if the current branching program has a high error rate then the split will significantly reduce the uncertainty (but not necessarily the error rate).

LEMMA 7. *Suppose that in ABPBA, for some $t \ge 1, \tau \ge 0$, $\text{err}(\mathcal{B}(\mathcal{L}_{t-1})) \ge \text{opt}(D, C) + \alpha + \tau$. Then after performing the split, the new partition $\mathcal{L}'_t$ satisfies $U(\mathcal{L}'_t, D) \le U(\mathcal{L}_t, D) - 4\gamma^2\tau/|\mathcal{L}_t|$.*

The next lemma bounds the increase due to a merge of two leaves as a function of their probabilities and uncertainties.

LEMMA 8. *For any disjoint $\ell_1, \ell_2 \subseteq X$ with $p_{\ell_1} \le p_{\ell_2} \le 1/2$ (or $p_{\ell_1} \ge p_{\ell_2} \ge 1/2$),*

$$\Delta(\ell_1, \ell_2) \le 2(w_{\ell_1} + w_{\ell_2})|u_{\ell_1} - u_{\ell_2}|.$$

The following lemma shows that if the branching program is large enough then there is a merge which will have a small increase in the uncertainty.

```
Input:
        access to $(m,\alpha,\gamma)$-weak agnostic learner $\mathcal{A}$
        access to distribution $D$ (estimated by iid samples)

Notation: Given a partition $\mathcal{L}$ (i.e., a set of leaves), for every $\ell \in L$, let $w_\ell = \Pr_D[x \in \ell]$, $p_\ell = \Pr_D[y=1|x \in \ell]$, $u_\ell = 2\sqrt{p_\ell(1-p_\ell)}$, $D|_\ell$ is the distribution obtained by conditioning on $x \in \ell$, and $\widehat{D|_\ell}$ is the balanced distribution of $D|_\ell$.

Algorithm:
  1. Start with the trivial partition $\mathcal{L}_0 = \{X\}$. (The branching program is a single leaf.)
  2. FOR $t := 1, 2, \ldots$,
     (a) Construct candidate splits: For each leaf $\ell \in \mathcal{L}_{t-1}$, run the weak learning algorithm $\mathcal{A}$ on the balanced distribution $\widehat{D|_\ell}$. The output $h_\ell : \ell \to \{0,1\}$ determines split $\ell^0, \ell^1$, where $\ell^i = \{x \in \ell | h_\ell(x) = i\}$, as well as $p_{\ell^0}$, $p_{\ell^1}$, $w_{\ell^0}$, and $w_{\ell^1}$.
     (b) Choose best split: Let $\lambda$ be the leaf in $\mathcal{L}$ that maximizes $\Delta(\lambda^0, \lambda^1)$. Let
             $\mathcal{L}'_t := \{\ell^0, \ell^1\} \cup \mathcal{L}_{t-1} \setminus \ell.$
         In the corresponding branching program label node $\lambda$ by $h_\lambda$ and add leaves $\lambda^0$ and $\lambda^1$.
     (c) Consider candidate merge: Sort the leaves $\mathcal{L}'_t = \{\ell_1, \ell_2, \ldots\}$ so that $p_{\ell_1} \le p_{\ell_2} \le \ldots$. Choose $i$ to minimize $\Delta(\ell_i, \ell_{i+1})$, i.e., the candidates to merge are the two consecutive leaves that minimize the increase in uncertainty.
     (d) Merge if safe: IF $\Delta(\ell_i, \ell_{i+1}) \le \Delta(\ell^0, \ell^1)/2$ THEN:
         • $\mathcal{L}_t := \{\ell_i \cup \ell_{i+1}\} \cup \mathcal{L}'_t \setminus \{\ell_i, \ell_{i+1}\}$ (In the corresponding branching program merge the leaves $\ell_i$ and $\ell_{i+1}$ to one leaf that receives the union of the incoming edges.)
         • ELSE $\mathcal{L}_t := \mathcal{L}'_t$.
```

Figure 1: The Agnostic Branching Program Boosting Algorithm (ABPBA)

```
Input: integers $a, b$ (such that $n = ab$), and
        $x_1, x_2, \ldots, x_m \in \{0,1\}^n$
Output: Disjoint $I_1, I_2, \ldots, I_{m'} \subseteq [m]$, for some $m'$.

Initialize: $I_1^1 = \{1\}, \ldots, I_m^1 = \{m\}$ and $\mathcal{I}^1$ be the collection of all $I_j^1$. Denote $x(I) = \sum_{k \in I} x_k$, for $I \subset \{1, \ldots, m\}$.

  • For $t = 1, \ldots, a$:
     1. For each $z \in \{0,1\}^b$ let $\mathcal{I}_z^t$ include all sets $I_j^t$ such that $x(I_j^t) = 0^{(t-1)b}zy$, for some $y$. Let $\mathcal{I}^t$ be the union of $\mathcal{I}_z^t$ over $z \in \{0,1\}^b$.
     2. For $z \in \{0,1\}^b$:
        (a) Randomly match the sets in $\mathcal{I}_z^t$ into pairs (if their number is odd, ignore one at random). Let $M_z^t$ be the matched pairs.
        (b) For each matched pair, $(j,k) \in M_z^t$, create a new set $I_i^{t+1} = I_j^t \cup I_k^t$.
  • output $\mathcal{I}^{a+1}$.
```

Figure 2: Grouping subroutine

```
Input:  $a, b$, $(x_1, y_1), \ldots, (x_m, y_m) \in \{0,1\}^n \times \{0,1\}$, and $x \in \{0,1\}^n$.
Output:  $\hat{y} \in \{0,1\}$.

1.  Run Grouping subroutine on $a, b, \langle x_1, \ldots, x_m, x \rangle$.
2.  If $m+1 \in I_i$ for some $i$, then output $\hat{y} = \sum_{j \in I_i; j \ne m+1} y_j$ else $\hat{y} = 0$.
```

Figure 3: Parity helper

Lemma 9. *Given a partition $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_L\}$, where $p_{\ell_1} \le p_{\ell_2} \le \ldots \le p_{\ell_L}$, there exists $1 \le i < L$ such that $\Delta(\ell_i, \ell_{i+1}) \le 144/L^2$.*

Combining the above lemmas we derive the following theorem.

Theorem 10. *Let $\epsilon > 0$ and consider an (m, α, γ)-weak agnostic learner. For some $t = O(\epsilon^{-1}\gamma^{-4})$ the error of the branching program generated by idealized ABPBA is at most* $\mathrm{opt} + \alpha + \epsilon$.

5. AGNOSTIC PARITY LEARNING

In this section we show that a variation of an algorithm given by Blum *et al* [4] is an agnostic weak learner for the class of parity functions on n bits. (See Section 2 for a definition of the class of parity functions.) The key ingredient of their algorithm is a procedure that given a point $x \in \{0,1\}^n$, finds a set I consisting of $O(\sqrt{n})$ out of the $2^{O(n/\log n)}$ points sampled by the example oracle, such that the sum (modulo 2) of the points in I is x. Then, since the noise in [4] is random, the sum (modulo 2) of the labels in I is weakly correlated with $c(x)$. This implies that the amplification in [4] can be done by repeating this experiment enough times and taking the majority, thus getting a signal which is strongly correlated with $c(x)$.

In contrast to the random noise model, the agnostic setting is much more challenging. In the agnostic model we consider adversarial noise, which might "know" which points we prefer in order to construct I, and plant noise on those points. For example, the distribution D might give very low weight to some prefixes, so the assumption that any prefix is equally likely does not hold anymore. For this reason, although our weak learner is quite similar to [4], our weak learner uses additional randomness to overcome the adversarial agnostic setting.

5.1 Weak agnostic parity learner

We first think of the n bits as a blocks of b bits (later, we will fix $a = \frac{\log n}{1000}$ and $b = 1000\frac{n}{\log n}$). The Grouping Subroutine, see Fig. 2 can be viewed as a randomized version of [4]. It takes a list of examples and partitions them into a number of groups of examples (leaving a small number out) such that the sum of examples in each group is the all 0 vector. The algorithm works in a manner similar to Gaussian elimination, but going b bits at a time. To zero the t-th block of b bits, it partitions the examples into 2^b groups based on their t-th block of bits, for $t = 1, 2, \ldots, a$. It then randomly pairs examples that are in the same group and replaces each pair by its sum (leaving the odd example out, if necessary). After doing this on each consecutive block, all that remains are some number m' of all-0 vectors, where $m/2^a - a2^b \le m' \le m/2^a$. The algorithm outputs the sets $I_1, \ldots, I_{m'} \subseteq [m]$ of indices used to form the zero vectors. (This means that the algorithm must keep track, not only of the sums of the vectors but which vectors were used in forming the sum.)

Based on the Grouping subroutine, we define the Parity helper procedure (Fig. 3) that, given m labeled examples and an unlabeled example x, predicts the label of x.

Lemma 11. *For any distribution D over $(x, y) \in \{0,1\}^n \times \{0,1\}$, let $(x_1, y_1), \ldots, (x_m, y_m), (x, y)$ be drawn independently from D. Let a, b, m be such that $2^{-a}m - a2^b \ge 3^{a+1}2^{b+1}$. Let $\hat{y}$ be the output of the* Parity helper *on $(x_1, y_1), \ldots, (x_m, y_m), x$. Then,*

$$\Pr[\hat{y} \ne y] \le \frac{1 - (1 - 2\,\mathrm{opt})^{2^a}}{2} + \frac{3^{a+1}2^b}{2^{-a}m - a2^b}.$$

Note that the probability above is over the random examples drawn from D as well as the randomness used by the algorithm. Note that the second expression can be made arbitrarily small, by setting m large enough. The proof of the above lemma is deferred to Appendix B. The main theorem, Theorem 2, follows as a corollary of Lemma 11 and the agnostic boosting theorem (Theorem 3).

Proof of Theorem 2. We claim the Parity helper, run with $a = \frac{\log n}{1000}$, $b = n/a$ is a (m, α, γ)-weak agnostic guesser for $\alpha = 2^{-n^{0.99}}$, $\gamma = 2^{n^{0.001} - n^{0.991} - 2}$, and $m = 4 \cdot 3^{a+1}2^b(2\alpha)^{-2^a}2^a + a2^{a+b} = 2^{O(n/\log n)}$. This is verified by plugging $\mathrm{opt} \le \frac{1}{2} - \alpha$ into the RHS of the quantity in Lemma 11, giving,

$$\frac{1 - 2^{(1-n^{0.99})n^{0.001}}}{2} + \frac{3^{a+1}2^b}{2^{-a}m - a2^b} \le \frac{1}{2} - 2^{(1-n^{0.99})n^{0.001} - 2}.$$

By Lemma 4, with a poly$(1/\gamma)$-factor blowup (in terms of the runtime and number of samples required), we can convert this to a weak agnostic learner. Finally, by Theorem 3, we can achieve error arbitrarily close to $\mathrm{opt} + \alpha$ in time $2^{O(n/\log n)}$. □

6. CONCLUSIONS

We have shown that the boosting by branching programs algorithm [15] can be analyzed in the agnostic setting, with the appropriate definition of agnostic weak learner. We have illustrated the utility of this fact in the first nontrivial algorithm for agnostically learning parity.

We remark that the algorithm of Blum *et al* [4] has been shown to have applications to a variety of other problems, such as determining the shortest lattice vector [1]. It would be very interesting to see if our agnostic algorithm has other applications. This paper helps us understand the actual difficulty of agnostic learning by showing that full agnostic boosting, is indeed possible, despite previous lower bounds.

Acknowledgments. We are very grateful to the anonymous referees for the accurate and pointed comments. We are also grateful to Rocco Servedio for allowing us to reproduce parts of the analysis of the earlier boosting algorithm.

7. REFERENCES

[1] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the 33rd Annual ACMSymposium on Theory of Computing*, 2001.

[2] Shai Ben-David, Philip M. Long, and Yishay Mansour. Agnostic boosting. In *Proceedings of the 14th Annual Conference on Computational Learning Theory*, pages 507–516, 2001.

[3] A. Blum, M. Furst, M. Kearns, and R. Lipton. Cryptographic Primitives Based on Hard Learning Problems. In *Advances in Cryptology – CRYPTO '93*, pages 278–291, 1993.

[4] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[5] V. Feldman, P. Gopalan, S. Khot, and A. K. Ponnuswami. New results for learning noisy parities and halfspaces. In *Proc. 47th IEEE Symp. on Foundations of Computer Science*, 2006.

[6] Dmitry Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *Journal of Machine Learning Research*, 4:101–117, 2003.

[7] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: the highly noisy case. *SIAM J. of Discrete Mathematics*, 13(4):535–570, 2000.

[8] David Haussler, Michael J. Kearns, Nick Littlestone, and Manfred K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, 1991.

[9] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In *ASIACRYPT*, pages 52–66, 2001.

[10] Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In *CRYPTO*, pages 293–308, 2005.

[11] A. Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. In *Proceedings of the 46th IEEE Symp. on Foundations of Computer Science*, 2005.

[12] A. Kalai and R. Servedio. Boosting in the presence of noise. To appear in *Journal of Computer and System Sciences*, 2005.

[13] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

[14] M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the Twenty-Eighth Annual Symposium on Theory of Computing*, pages 459–468, 1996.

[15] M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58(1):109–128, 1999.

[16] M. Kearns, R. Schapire, and L. Sellie. Toward Efficient Agnostic Learning. *Machine Learning*, 17(2/3):115–141, 1994.

[17] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *APPROX–RANDOM*, pages 378–389, 2005.

[18] Y. Mansour and D. McAllester. Boosting using branching programs. *Journal of Computer and System Sciences*, 64(1):103–112, 2002.

[19] E. Mossel and S. Roch. Learning nonsingular phylogenies and hidden markov models. In *To appear in Proceedings of the 37th Annual Symposium on Theory of Computing (STOC)*, 2005.

[20] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM Symposium on Theory of Computing (STOC-05)*, pages 84–93, New York, 2005.

[21] R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[22] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

APPENDIX

A. PROOF OF LEMMA 4

Let $h_i : X \to \{0,1\}$ be the function computed by circuit Q_i, and h be the function of the output circuit. The error of L is a non-negative random variable with mean at most $\frac{1}{2} - \gamma$. Since $\text{err}(h_i) \geq 0$, we have that,

$$\text{P}\left[\text{err}(h_i) \geq \frac{1}{2} - \frac{\gamma}{2}\right] \leq \frac{1/2-\gamma}{1/2-\gamma/2} = \frac{1-2\gamma}{1-\gamma} \leq 1-\gamma.$$

Hence, if we execute the algorithm $1/\gamma$ times, with probability at most $(1-\gamma)^{1/\gamma} \leq 1/e$, we will have no hypothesis with error $\leq 1/2 - \gamma/2$. In the other case, we argue that the reduction succeeds in finding one with error at most $1/2-\gamma/4$, with probability $\geq .9$. Hence, the failure probability of the algorithm is, by the union bound, at most $1/e + .1 \leq 1/2$.

By Chernoff bounds, the probability that hypothesis h has empirical error more than $\gamma/8$ from its true error, on a held out test set of size $\frac{32}{\gamma^2}\log\frac{20}{\gamma}$ is at most $2e^{-2(\gamma/8)^2\frac{32}{\gamma^2}\log\frac{20}{\gamma}} \leq .1\gamma$. If this happens for all $1/\gamma$ hypotheses, then we will find one with error at most $1/2-\gamma/4$. By the union bound, we will fail on this part with probability at most $.1\gamma(1/\gamma) \leq .1$. $\square$

B. PARITY ANALYSIS

It is easy to see that the GROUPING subroutine partitions the examples into a number of groups which each sum to 0 (and leaves out a small number of examples). Let $m_0 = 2^{-a}m - a2^b$. We will be interested in the cases of m, a, b where $m_0 \gg 1$.

LEMMA 12. *For any integers $a, b \geq 1, n = ab$, and any m unlabeled examples $x_1, x_2, \ldots, x_m \in \{0,1\}^n$, the* GROUPING *subroutine outputs disjoint sets $I_1^{a+1}, \ldots, I_{m'}^{a+1}$ such that, for each $j \leq m'$, $x(I_j^{a+1}) = 0$. Also, $m_0 \leq m' \leq 2^{-a}m$.*

Fix distribution D on (x, y). Fix a parity function $c \in P_n$ that has minimal error on distribution D, so $\text{err}_{P_n}(c) = \text{opt}_{P_n}(D)$. (Note that c is not necessarily unique, and that different minimal error functions can be very far from each other on D. This is part of the challenge of the agnostic model.) An example (x, y) is called *noisy* if $y \neq c(x)$ and *quiet* otherwise. Let $\langle(x_i, y_i)\rangle_{i=1}^m$ be a data set of labeled examples chosen independently from D. Let η be the random variable that denotes the fraction of the data that are noisy, so $\text{E}[\eta] = \text{opt}$. It will also be convenient to talk about the *advantage* $\alpha = (1-\eta) - \eta = 1 - 2\eta \in [-1, 1]$, the fraction of quiet examples minus noisy examples.

Consider passing the unlabeled examples to the GROUPING subroutine. We say that I_j^t is quiet (resp. noisy) if an even (resp. odd) number of the corresponding labeled examples (x_i, y_i), $i \in I_j^t$, are noisy. Similarly, we define η^t, η_z^t to be the empirical fraction of noisy sets in $\mathcal{I}^t$ and $\mathcal{I}_z^t$, respectively. Also, let $\alpha^t = 1 - 2\eta^t$ and $\alpha_z^t = 1 - 2\eta_z^t$.

The key, perhaps surprising, technical lemma is the following.

LEMMA 13. *For any $t = 1, 2, \ldots, a$,*

$$\text{E}[\alpha^{t+1}] \geq \text{E}[\alpha^t]^2 - \frac{6(2^b)}{m_0}.$$

In Appendix C and D, we give the proofs of Lemma 11 and Lemma 13. Next, we give some intuition about why Lemma 13 is true.

Informal Intuition for Lemma 13

Our analysis will show that worst case is approximately the case of classification noise, where D is the distribution in which each label y disagrees with $c(x)$ with probability opt, independent of x. This is not precisely true, but classification noise will be close to the worst case.

To gain some intuition about why this is the case, let us first be very imprecise. For this section only, let's ignore various roundoff errors and assume that everything works out exactly according to its expectation, e.g., $\eta = \text{opt}$. (However, we are not assuming that there is random classification noise and we have an arbitrary distribution D.)

Given η_z^t, let us estimate the fraction of noisy examples in M_z^t, for some $z \in \{0,1\}^b$. A random pair has probability approximately $\eta_z^t(1-\eta_z^t) + (1-\eta_z^t)\eta_z^t$ of being noisy. (This is not exact, because after the first example in a pair is chosen to be, say, noisy , the chance that the second one is quiet is larger due to the fact that pairs are drawn without replacement.) Hence, we expect the advantage (the fraction of quiet minus noisy examples) on M_z^t to be about,

$$1 - 2(\eta_z^t(1-\eta_z^t) + (1-\eta_z^t)\eta_z^t) = (1-2\eta_z^t)^2 = (\alpha_z^t)^2.$$

So the advantage roughly squares on each set. Hence the advantage on $\mathcal{I}^{t+1}$, which is just a weighted average of advantages on $\mathcal{I}_z^t$, over z, is about:

$$\alpha^{t+1} \approx \sum_{z\in\{0,1\}^b} \frac{|\mathcal{I}_z^t|}{|\mathcal{I}^t|}(\alpha_z^t)^2 \le \left(\sum_{z\in\{0,1\}^b} \frac{|\mathcal{I}_z^t|}{|\mathcal{I}^t|}\alpha_z^t\right)^2 = (\alpha^t)^2.$$

The inequality above follows from the convexity of the function $g(z) = z^2$. In the case of random classification noise, the inequality is approximately an equality.

C. PROOF OF LEMMA 13

Let us fix the state S^t of the algorithm at the beginning of iteration t. Hence, the data set, $\mathcal{I}_z^t$, and $\alpha_z^t = 1 - 2\eta_z^t$ are known for all z. The algorithm still makes random choices that determine the matchings M_z^t in $\mathcal{I}_z^t$. For any one z, note that $|M_z^t| = \lfloor \frac{|\mathcal{I}_z^t|}{2} \rfloor$. A pair $(j,k) \in M_z^t$ is noisy if $x(I_j^t \cup I_k^t)$ is noisy. The chance that a pair in M_z^t is noisy can be computed as follows. For a pair to be noisy one element has to be noisy and the other quite. The first of these two sets is noisy with probability η_z^t and the chance that the second one is quiet, given that the first was noisy is $\frac{(1-\eta_z^t)|\mathcal{I}_z^t|}{|\mathcal{I}_z^t|-1}$, since the two are drawn without replacement. Similarly, the probability that the first set is quiet and the second is noisy is $(1-\eta_z^t)\frac{\eta_z^t|\mathcal{I}_z^t|}{|\mathcal{I}_z^t|-1}$, which is the same. Hence, the expected number of noisy sets in M_z^t is:

$$\begin{aligned} 2\eta_z^t(1-\eta_z^t)\frac{|\mathcal{I}_z^t|}{|\mathcal{I}_z^t|-1}\left\lfloor\frac{|\mathcal{I}_z^t|}{2}\right\rfloor &\le \eta_z^t(1-\eta_z^t)(|\mathcal{I}_z^t|+2) \\ &\le \eta_z^t(1-\eta_z^t)|\mathcal{I}_z^t|+1. \end{aligned}$$

In the above, we have used the facts that $\eta_z^t(1-\eta_z^t) \le 1/2$ and

$$2\frac{j}{j-1}\left\lfloor\frac{j}{2}\right\rfloor \le \frac{j^2}{j-1} = j+1+\frac{1}{j-1} \le j+2$$

for any integer $j \ge 2$. For $j = 0$ or 1, we take $0/0 = 0$ and the inequality holds as well.

Let S^t denote the state of the algorithm at the start of iteration t, including previously used random bits but not including randomness used on period t. Hence, we have that the *expected* number of noisy examples, given the state at time t, is:

$$\mathrm{E}[|\mathcal{I}^{t+1}|\eta^{t+1} \mid S^t] \le \sum_{z\in\{0,1\}^b} (\eta_z^t(1-\eta_z^t)|\mathcal{I}_z^t|+1).$$

In other words, the above is the expectation over the randomness that determines the algorithms decisions on iteration t. Recall that $|\mathcal{I}^t| \ge m_0$ (from Lemma 12). Since $|\mathcal{I}^t| \le 2|\mathcal{I}^{t+1}| + 2^b$, we have,

$$\begin{aligned} \mathrm{E}[|\mathcal{I}^t|\eta^{t+1} \mid S^t] &\le 2\mathrm{E}[|\mathcal{I}^{t+1}|\eta^{t+1} \mid S^t] + 2^b \\ &\le 2\sum_{z\in\{0,1\}^b} (\eta_z^t(1-\eta_z^t)|\mathcal{I}_z^t|+1) + 2^b \\ &= 2\sum_{z\in\{0,1\}^b} \eta_z^t(1-\eta_z^t)|\mathcal{I}_z^t| + (3)2^b \end{aligned}$$

Since $\mathcal{I}^t$ is fixed, given S^t, we can divide by $|\mathcal{I}^t|$.

$$\begin{aligned} \mathrm{E}[\eta^{t+1} \mid S^t] &\le \sum_{z\in\{0,1\}^b} \frac{|\mathcal{I}_z^t|}{|\mathcal{I}^t|}2\eta_z^t(1-\eta_z^t) + \frac{(3)2^b}{m_0} \\ &= \sum_{z\in\{0,1\}^b} \frac{|\mathcal{I}_z^t|}{|\mathcal{I}^t|}2\frac{1-\alpha_z^t}{2}\frac{1+\alpha_z^t}{2} + \frac{(3)2^b}{m_0} \\ &= \sum_{z\in\{0,1\}^b} \frac{|\mathcal{I}_z^t|}{|\mathcal{I}^t|}\frac{1-(\alpha_z^t)^2}{2} + \frac{(3)2^b}{m_0} \\ &= \frac{1}{2} - \sum_{z\in\{0,1\}^b} \frac{|\mathcal{I}_z^t|}{|\mathcal{I}^t|}\frac{(\alpha_z^t)^2}{2} + \frac{(3)2^b}{m_0} \end{aligned}$$

Since, by definition, $\eta^{t+1} = \frac{1}{2}(1-\alpha^{t+1})$, we have

$$\mathrm{E}[\alpha^{t+1} \mid S^t] \ge \sum_{z\in\{0,1\}^b} \frac{|\mathcal{I}_z^t|}{|\mathcal{I}^t|}(\alpha_z^t)^2 - \frac{(6)2^b}{m_0}$$

However, by convexity of the function $g(x) = x^2$, we have,

$$\sum_{z\in\{0,1\}^b} \frac{|\mathcal{I}_z^t|}{|\mathcal{I}^t|}(\alpha_z^t)^2 \ge \left(\sum_{z\in\{0,1\}^b} \frac{|\mathcal{I}_z^t|}{|\mathcal{I}^t|}\alpha_z^t\right)^2 = \left(\alpha^t\right)^2$$

Since the last expectation held for any S^t, it holds in expectation over all S^t, i.e.,

$$\mathrm{E}[\alpha^{t+1}] \ge \mathrm{E}[(\alpha_z^t)^2] - \frac{(6)2^b}{m_0}.$$

In the above, expectations are taken over all randomness in the input and algorithm. Finally, using the fact that $\mathrm{E}[X^2] \ge \mathrm{E}[X]^2$ (again by convexity of $g(x) = x^2$ or non-negativity of variance) we have the lemma. □

D. REMAINING PROOFS

Proof of Lemma 6. WLOG $p = \mathrm{P}_D[y=1] \le 1/2$, so we need to show that $1/2 - \mathrm{opt}_{\widehat{D}}(C) \ge p - \mathrm{opt}_D(C)$.

Fix any $c \in C$. Let $a = \mathrm{P}[c(x) = 0|y = 1]$ and $b = \mathrm{P}[c(x) = 1|y = 0]$ so that $\mathrm{err}_D(c) = pa + (1-p)b$ and

$\mathrm{err}_{\widehat{D}}(c) = (a+b)/2$. To complete the lemma, it suffices to show that,

$$\begin{aligned}\frac{1}{2} - \frac{a+b}{2} &\geq p - (pa + (1-p)b) \iff \\ 0 &\geq \left(p - \frac{1}{2}\right)(1 - a + b).\end{aligned}$$

The above is clearly true since $b \geq 0$, $a \leq 1$ and $p \leq 1/2$. $\square$

Proof of Lemma 7. Call a leaf ℓ *hazy* if

$$\min(p_\ell, 1-p_\ell) - \underset{D|_\ell}{\mathrm{opt}}(C) \geq \alpha,$$

and let $\mathcal{H}$ be the set of hazy leaves in $\mathcal{L}_t$. For each hazy leaf $\ell \in \mathcal{H}$, by Lemma 6, $\mathrm{opt}_{\widehat{D|_\ell}}(C) \leq 1/2 - \alpha$, and hence an (m, α, γ)-weak agnostic learner would return an hypothesis h_ℓ, for the balanced distribution, with error at most $1/2 - \gamma$. By Equation (1), splitting such a leaf would reduce the uncertainty by at least $2\gamma^2 w_\ell u_\ell$. It remains to show that there is a leaf $\ell \in \mathcal{H}$ such that $w_\ell u_\ell \geq 2\tau/|\mathcal{L}_t|$.

Since the error is $\mathrm{err}(\mathcal{B}(\mathcal{L}_{t-1})) = \sum_\ell w_\ell \min(p_\ell, 1-p_\ell) \geq \mathrm{opt}_D(C) + \alpha + \tau$, and $\mathrm{opt}_D(C) \geq \sum_\ell w_\ell \,\mathrm{opt}_{D|_\ell}(C)$, we have,

$$\sum_\ell w_\ell \left(\min(p_\ell, 1-p_\ell) - \underset{D|_\ell}{\mathrm{opt}}(C)\right) \geq \alpha + \tau.$$

Since for each $\ell \notin \mathcal{H}$, $\min(p_\ell, 1-p_\ell) - \mathrm{opt}_{D|_\ell}(C) \leq \alpha$, we have, $\sum_{\ell \notin \mathcal{H}} w_\ell(\min(p_\ell, 1-p_\ell) - \mathrm{opt}_{D|_\ell}(C)) \leq \alpha$. Therefore,

$$\sum_{\ell \in \mathcal{H}} w_\ell \left(\min(p_\ell, 1-p_\ell) - \underset{D|_\ell}{\mathrm{opt}}(C)\right) \geq \tau.$$

Furthermore, since $u_\ell \geq 2\min(p_\ell, 1-p_\ell)$, we have $\sum_{\ell \in \mathcal{H}} w_\ell u_\ell \geq 2\tau$. Since there are at most $|\mathcal{L}_t|$ leaves in $\mathcal{H}$, one of them must have $w_\ell u_\ell \geq 2\tau/|\mathcal{L}_t|$. Hence, applying the split on this leaf would give a reduction in uncertainty of at least $4\gamma^2\tau/|\mathcal{L}_t|$. $\square$

Proof of Lemma 8. For the purposes of this proof let $w_i = w_{\ell_i}$, $p_i = p_{\ell_i}$, $q_i = 1 - p_i$, $u_i = u_{\ell_i} = 2\sqrt{p_i q_i}$, $\ell = \ell_1 \cup \ell_2$, $p = p_\ell$, $q = 1-p$, $u = u_\ell$, and $w = w_\ell$. Since $p_{\ell_1} \leq p_{\ell_2} \leq 1/2$ we have that $u_1 \leq u \leq u_2$. Then,

$$\begin{aligned}\frac{1}{2}\Delta(\ell_1, \ell_2) &= wu - w_1 u_1 - w_2 u_2 \\ &= w\left[\frac{w_1}{w}(u - u_1) + \frac{w_2}{w}(u_2 - u)\right] \\ &\leq w(u_2 - u_1)\end{aligned}$$

$\square$

Proof of Lemma 9. Let $p_i = p_{\ell_i}$, $u_i = u_{\ell_i}$ and $w_i = w_{\ell_i}$. By symmetry, WLOG we may suppose that $p_{L/2} \leq 1/2$. We have $0 \leq u_1 \leq u_2 \leq \ldots \leq u_{L/2} \leq 1$. Set $x_i = (w_i + w_{i+1})/2$ and $y_i = |u_{i+1} - u_i|$. Then $\sum_{i=1}^{L/2-1} x_i \leq 1$ and $\sum_{i=1}^{L/2-1} y_i \leq 1$ Therefore, the number of x_i such that $x_i \leq 6/L$ is at least $(1/2 - 1/6)L$. Similarly, the number of y_i such that $y_i \leq 6/L$ is at least $(1/2 - 1/6)L$. Hence, there is an index i such that $x_i \leq 6/L$ and $y_i \leq 6/L$. This implies that there is an index $i \leq L/2 - 1$ such that $\frac{w_i + w_{i+1}}{2}|u_{i+1} - u_i| \leq 36/L^2$. By Lemma 8 we have that $\Delta(\ell_i, \ell_{i+1}) \leq 144/L^2$. $\square$

Proof of Theorem 10. Let an iteration be a sequence of a split operation followed by a possible merge operation. First, note that by the definition of the ABPBA algorithm, the value of $U(\mathcal{L}, D)$ cannot increase each iteration. Let phase k be the iterations starting at the first time in which $\frac{1}{2}U(\mathcal{L}, D) < \mathrm{opt} + \alpha + 2^{-k}$ and ending just before $\frac{1}{2}U(\mathcal{L}, D) < \mathrm{opt} + \alpha + 2^{-(k+1)}$. Recall that half the uncertainty is a lower-bound on the error of the resulting branching program.

We claim that during phase k, there cannot be more than $L_k = 144\gamma^{-2}2^k$ leaves. The reason is that, once there become L_k leaves, by Lemma 7 the uncertainty $U \geq \mathrm{opt} + \alpha + 2^{-(k+1)}$ prior to the split decreases by at least $2^{1-k}\gamma^2/L_k$. Lemma 9 then implies that there is some merge which would increase the uncertainty by at most $144/L_k^2 \leq \frac{1}{2}2^{1-k}\gamma^2/L_k$, for our choice of L_k. Thus this merge will be performed in Step 7 and there will again be L_k leaves.

Thus the net reduction in uncertainty is at least $2^{-k}\gamma^2/L_k$ during each iteration of phase k. Since the uncertainty drops by at most 2^{-k} during phase k, it can last at most $L_k\gamma^{-2}$ iterations. Thus phase $r = \lceil \log(1/\epsilon) \rceil$ will be reached after at most $\sum_{k=1}^r L_k \gamma^{-2}$ iterations, which is bounded by $O(\gamma^{-4}\epsilon^{-1})$. $\square$

Proof of Lemma 12. Each $x(I_j^t)$ has the first $t-1$ blocks all 0, for all $t \geq 1$, so $x(I_j^{a+1})$ consists of only 0. Since each example contributes to at most one I_j^{a+1}, the sets I_j^{a+1} are disjoint. Since each set consists of 2^a examples, there can be at most $2^{-a}m$ such sets. Let n^t be the number of different sets I_j^t, i.e., $|\mathcal{I}^t|$. Note that $n^0 = m + 1$ and $n^{a+1} = m'$. Since we throw out at most 2^b vectors during each iteration (one per possible value z of b bits),we have $n^{t+1} \geq n^t/2 - 2^b$, implying $n^{a+1} \geq 2^{-a}m - a2^b$. $\square$

Proof of Lemma 11. First, we have $\mathrm{E}[\alpha^1] = 1 - 2\,\mathrm{opt}$. We claim that for each $t = 0, 1, \ldots, a$,

$$\mathrm{E}[\alpha^{t+1}] \geq (1 - 2\,\mathrm{opt})^{2^t} - \frac{(6)3^t 2^b}{m_0}. \tag{2}$$

We argue this by induction on t. The base case $t = 0$ is true by definition. For $t+1$, from Lemma 13, we have $\mathrm{E}[\alpha^{t+1}] \geq \mathrm{E}[\alpha^t]^2 - \frac{6(2^b)}{m_0}$. Since for $\epsilon, z \in [0,1]$, we have $z^2 - 2\epsilon \leq (z - \epsilon)^2$. Therefore,

$$\begin{aligned}\mathrm{E}[\alpha^{t+1}] &\geq \mathrm{E}[\alpha^t]^2 - \frac{6(2^b)}{m_0} \\ &\geq \left((1 - 2\,\mathrm{opt})^{2^{t-1}} - \frac{(6)3^{t-1}2^b}{m_0}\right)^2 - \frac{6(2^b)}{m_0} \\ &\geq (1 - 2\,\mathrm{opt})^{2^t} - 2\frac{(6)3^{t-1}2^b}{m_0} - \frac{6(2^b)}{m_0} \\ &\geq (1 - 2\,\mathrm{opt})^{2^t} - \frac{(6)3^t 2^b}{m_0}\end{aligned}$$

Since the quantity we want to bound is $\eta^{a+1} = \frac{1 - \alpha^{a+1}}{2}$, rearranging terms gives the lemma. $\square$