Boosting in the Presence of Noise

Adam Tauman Kalai[*]
Toyota Technological Institute at Chicago
Chicago, Illinois 60637
kalai@tti-c.org

Rocco A. Servedio[†]
Department of Computer Science
Columbia University
New York, NY 10027
rocco@cs.columbia.edu

Abstract

Boosting algorithms are procedures that "boost" low accuracy weak learning algorithms to achieve arbitrarily high accuracy. Over the past decade boosting has been widely used in practice and has become a major research topic in computational learning theory. In this paper we study boosting in the presence of random classification noise, giving both positive and negative results.

We show that a modified version of a boosting algorithm due to Mansour and McAllester [15] can achieve accuracy arbitrarily close to the noise rate. We also give a matching lower bound by showing that no efficient black-box boosting algorithm can boost accuracy beyond the noise rate (assuming that one-way functions exist). Finally, we consider a variant of the standard scenario for boosting in which the "weak learner" satisfies a slightly stronger condition than the usual weak learning guarantee. We give an efficient algorithm in this framework which can boost to arbitrarily high accuracy in the presence of classification noise.

[*] Work done while the author was at the Laboratory for Computer Science, MIT, and was supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

[†] Work done while the author was at the Division of Engineering and Applied Sciences, Harvard University, and supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship and by NSF grant CCR-98-77049.

1 Introduction

In Valiant's Probably Approximately Correct (PAC) learning model, a successful learning algorithm must be able to achieve any arbitrarily low error rate given random examples drawn from any fixed probability distribution. In an early paper, Kearns and Valiant [13] proposed the notion of a *weak* learning algorithm which need only achieve some error rate bounded away from $\frac{1}{2}$, and posed the question of whether weak and strong learning are equivalent for efficient (polynomial time) learning algorithms. Soon afterward, in a celebrated result Schapire gave a positive answer to this question [16]. Schapire gave an efficient *boosting* algorithm which, given access to any weak learning algorithm, uses the weak learner to generate a hypothesis with arbitrarily low error. Since Schapire's initial result boosting has become one of the biggest successes of computational learning theory; boosting algorithms have been intensively studied from a theoretical perspective and are widely used in practice.

The standard PAC learning model assumes that all examples received by the learner are labeled correctly, i.e. the data has no noise. An important question, which was asked by Schapire in his original paper [16] and by several subsequent researchers [2], is whether it is possible to efficiently perform boosting in the presence of noise. Since real data is frequently noisy, this question is of significant practical as well as theoretical interest.

In this paper we give a detailed study of boosting in the presence of *random classification noise*. In the random classification noise model, the binary label of each example which the learner receives is independently flipped from the true label $f(x)$ with probability η for some fixed $0 < \eta < \frac{1}{2}$; the value η is referred to as the *noise rate*. Random classification noise is the most standard and widely studied noise model in learning theory. We give both positive and negative results for boosting in this model as described below.

1.1 Our Results

We first demonstrate that decision-tree-like boosting algorithms can boost accuracy arbitrarily close to the noise rate. In particular, we analyze a modified version of the "branching programs" booster of Mansour and McAllester [15], which built on a boosting analysis of decision trees due to Kearns and Mansour [11]. We refer to the boosting algorithm from [15] as the MM boosting algorithm, and to our *modified* version as the MMM boosting algorithm.

We next show that in general it is *not* possible to boost to any error rate lower than the noise rate using a "black-box" polynomial time boosting algorithm. This negative result assumes only that one-way functions exist. Some computational hardness assumption is required since in exponential time any weak learner can be boosted to arbitrary accuracy in the presence of noise. (Draw a polynomial size noisy data set, exhaustively guess which labels are noisy, and run a standard boosting algorithm.)

The results described above assume that the boosting algorithm has access to a weak learner as defined by Kearns and Valiant, i.e. an algorithm which, given examples drawn from a distribution $\mathcal{D}$, produces a hypothesis whose error rate relative to the target function is bounded away from 1/2. For our second positive result we consider a slightly stronger notion of an *okay* learner (precisely defined in Section 6) which produces a hypothesis whose *covariance* with the target function is bounded away from 0. We show that if the MMM boosting algorithm has access to an okay learner, then it can boost to achieve arbitrarily low error in the presence of random classification noise.

	noise	no noise
true positive example	$p\eta$	$p(1-\eta)$
true negative example	$(1-p)\eta$	$(1-p)(1-\eta)$

Table 1: Examples labeled 1 are either noisy negative examples or nonnoisy positive examples. Thus the frequency of true positive examples among examples labeled 1 is $\frac{p(1-\eta)}{p(1-\eta)+(1-p)\eta}$ which is less than $\frac{1}{2}$ if $p < \eta < \frac{1}{2}$.

1.2 Our approach

Recall that a weak learning algorithm must output a hypothesis with error at most $\frac{1}{2} - \gamma$ when given examples drawn from any distribution $\mathcal{D}$. A simple but useful observation is the following: if $\mathcal{D}$ is balanced between positive and negative examples then the hypothesis generated by a weak learner provides some useful information, but if $\mathcal{D}$ is unbalanced then the weak learner can output a trivial hypothesis and still satisfy the guarantee. For example, if $\gamma = 0.1$ and $\mathcal{D}$ puts probability weight 0.8 on positive examples, then the identically-1 hypothesis is a legitimate output for the weak learner. Thus the only way to ensure that a weak learner gives some useful information is to run it on a distribution which is roughly balanced between positive and negative examples. If the distribution $\mathcal{D}$ is unbalanced, then some sort of filtering or reweighting must be performed to obtain a balanced distribution $\mathcal{D}'$; all known boosting algorithms take this approach when given a constant weak hypothesis.

The main idea behind our negative result is that in the presence of classification noise, it can be difficult to obtain a balanced distribution $\mathcal{D}'$. Consider a scenario where $\mathcal{D}$ puts weight $p < \frac{1}{2}$ on positive examples. To make the weak learner do something useful, we would like to reweight to a balanced distribution $\mathcal{D}'$. Intuitively, the best way to do this is to discard some examples which are labeled 0. However, if $p < \eta$ then even among examples which are labeled 1, less than half are true positive examples (see Table 1). Thus we cannot construct a new distribution which forces the weak learner to do something useful, so we cannot boost to high accuracy. In Section 5 we make these ideas precise and give a hardness proof.

For our positive results we consider a modified version of the "branching program" boosting algorithm of Mansour and McAllester [15]. Our analysis exploits the fact that their scheme causes the (possibly noisy) label of a given example to play a relatively small role in its reweighting. This is in contrast with several other boosting algorithms, such as AdaBoost [6], in which a noisy label can cause an example to receive exponentially more weight than it would otherwise receive. We note that several researchers [3, 17] have empirically observed that standard boosting algorithms such as AdaBoost can perform poorly on noisy data, and indeed it has been suggested that this poor performance is due to AdaBoost's tendency to construct distributions which put a great deal of weight on a few noisy examples [3].

1.3 Related Work

The elegant Statistical Query model introduced by Kearns [10] is a model in which the learner does not receive labeled examples but instead can obtain estimates of statistical properties of the distribution of labeled examples. Aslam and Decatur gave an algorithm for boosting any Statistical Query weak learner to arbitrary accuracy [1]. Since every Statistical Query algorithm can be simulated using a noisy example oracle [10], their result seems to imply that any Statistical Query weak learning algorithm can be boosted even with noise.

However, Aslam and Decatur's result does not allow the Statistical Query weak learner to have access to unlabeled examples from the distribution, which is sometimes considered part of the Statistical Query model. In fact, the "unboostable" weak learning algorithm we present in Section 5 can be viewed as a Statistical Query algorithm that requires access to unlabeled examples. This suggests that it may be impossible, in general, to boost Statistical Query algorithms that have access to unlabeled examples, or that Aslam and Decatur's result may be the strongest possible.

One of the most impressive examples of noise-tolerant learning is that of learning a noisy half-space [2]. Their algorithm uses a special outlier-removal process that examines unlabeled points. Thus, while their algorithm is, in the broadest sense, a Statistical Query algorithm, Aslam and Decatur's boosting cannot be used directly on their approach. Instead, they give a special-case boosting approach for their problem.

In follow-up work, it has been shown that branching programs can be used to boost under a stronger model of noise [9]. The model considered there is an arbitrary distribution over $X \times Y$, where, for simplicity, say $Y = \{0, 1\}$. As in the p-concept mode [12] the goal is to learn $f(x) = E[y|x]$ for a random example (x, y) from the distribution, and the error of a hypothesis h is measured by $E[(h(x) - f(x))^2]$. It is shown that as long as one can find a hypothesis which is positively correlated (has a positive correlation coefficient) with the target function, boosting is possible. As an application, it is shown that the class of generalized additive models (with monotonic link functions), popular in the statistics literature, can be learned by such boosting.

The above model of "noise" is stronger and weaker in some senses. Its strength is that the noise is not necessarily uniform, and the hypothesis has to learn the noise as well. However, in the case of uniform classification noise very near 1/2, the constant hypothesis $h(x) = 1/2$ is quite accurate and real learning only has to be done to get very small error. In contrast, according to the standard definition of accuracy in a noisy setting, which is with respect to a noiseless test set, this high-noise case is more difficult.

2 PAC Learning Preliminaries

Our results are for the model of PAC learning in the presence of classification noise. For a detailed introduction to PAC learning see [14].

A *concept class* C is a class of Boolean functions over some *instance space* X. We assume throughout that the instance space X is of dimension n, i.e. $X = \mathbf{R}^n$ or $X = \{0, 1\}^n$, and we are interested in algorithms whose running time is polynomial in n (and other parameters).

Let f be a function in C, $\mathcal{D}$ a distribution over X, and η a value $0 \leq \eta < \frac{1}{2}$. A *noisy example oracle* is an oracle $EX(f, \mathcal{D}, \eta)$ which works as follows: each time $EX(f, \mathcal{D}, \eta)$ is invoked, it returns a labeled example $\langle x, b \rangle \in X \times \{0, 1\}$ where $x \in X$ is drawn from distribution $\mathcal{D}$ and b is independently chosen to be $f(x)$ with probability $1 - \eta$ and $1 - f(x)$ with probability η.

Let $f \in C$ be a fixed target function. A noise-tolerant PAC learning algorithm for a concept class C is an algorithm which has the following property: for any $\epsilon, \delta > 0$, any $0 \leq \eta < \frac{1}{2}$, any target function $f \in C$, and any distribution $\mathcal{D}$ over X, if the algorithm is given access to $EX(f, \mathcal{D}, \eta)$ then with probability $1 - \delta$ it outputs a hypothesis h such that $\Pr_{x \in \mathcal{D}}[h(x) \neq f(x)] < \epsilon$. We refer to $\Pr_{x \in \mathcal{D}}[h(x) \neq f(x)]$ as the *error* of h under $\mathcal{D}$.

A noise-tolerant weak learning algorithm is an algorithm which satisfies the PAC criterion only for sufficiently large ϵ. More precisely, we have:

Definition 1 *Let $0 < \gamma < \frac{1}{2}$. A noise-tolerant γ-weak learning algorithm for a concept class C is an algorithm A that takes inputs n, δ and is given access to a noisy example oracle $\mathcal{O}$, with* *the following property. For all n, δ, if $\mathcal{O}$ is a noisy example oracle $EX(f, \mathcal{D}, \eta)$ where $f \in C$, $\mathcal{D}$ is any distribution on $\{0,1\}^n$, and $0 \leq \eta < \frac{1}{2}$, then A runs in time poly$(n, \frac{1}{1-2\eta}, \frac{1}{\delta})$ and with probability at least $1-\delta$, A outputs a poly$(n, \frac{1}{\delta}, \frac{1}{\gamma}, \frac{1}{1-2\eta})$-time evaluable hypothesis h such that $\Pr_{x \in \mathcal{D}}[h(x) \neq f(x)] \leq \frac{1}{2} - \gamma$.*

A boosting algorithm is an algorithm which, given access to a weak learning algorithm, can generate a hypothesis h with arbitrarily low error. More precisely, we have:

Definition 2 *A black-box noise-tolerant booster is an algorithm B that is given access to an oracle $\mathcal{O}$ and black-box access to an algorithm A, with the following property. For all concept classes C, for all $0 < \gamma < \frac{1}{2}$, for all $0 \leq \eta < \frac{1}{2}$, for all n, ϵ, δ, we have: if A is a noise-tolerant γ-weak learning algorithm for C and $\mathcal{O}$ is a noisy example oracle $EX(f, \mathcal{D}, \eta)$ where $f \in C$ and $\mathcal{D}$ is any distribution on $\{0,1\}^n$, then $B^{\mathcal{O},A}$ runs in time poly$(n, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{\gamma}, \frac{1}{1-2\eta})$ and with probability at least $1-\delta$ B outputs a poly$(n, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{\gamma}, \frac{1}{1-2\eta})$-time evaluable hypothesis h such that $\Pr_{x \in \mathcal{D}}[h(x) \neq f(x)] \leq \epsilon$.*

We note that in both our positive and negative results, the boosting algorithm B calls the weak learning algorithm A as a black box; B may run A using any oracle $\mathcal{O}$ which B is able to provide, but B cannot "read the code" of A. Thus our negative results hold only for boosting algorithms which operate in this black-box way. We feel that this is a minor restriction to put on boosting algorithms since all known boosting algorithms (including the MM boosting algorithm which we analyze) work in a black-box way – they call the weak learner and use the hypotheses which it generates, but do not inspect the internal state of the weak learner during its execution.

3 MM: Noise-Free Boosting

In this section we describe a particular boosting algorithm and analyze its performance in the absence of noise (i.e. when $\eta = 0$). The algorithm we describe here is essentially the branching program booster of Mansour and McAllester [15] (which built on ideas from Kearns and Mansour's paper [11]), and we henceforth refer to it as the MM boosting algorithm. Our goal here is to set the stage for our analysis of the MMM algorithm (modified MM) in the presence of noise, which we give in Sections 4 and 6. Our presentation and analysis of the MM algorithm are slightly different from [15] in order to facilitate our presentation and analysis of the MMM algorithm in Sections 4 and 6.

3.1 Preliminaries

Throughout this section we let $f \in C$ be a fixed target function and $\mathcal{D}$ be a fixed distribution over X. For $\ell \subseteq X$ we write $\mathcal{D}|_\ell$ to denote $\mathcal{D}$ conditioned on $x \in \ell$, i.e. $\mathcal{D}|_\ell(S) = \Pr_{\mathcal{D}}[x \in S \mid x \in \ell]$. We write p_ℓ to denote $\Pr_{\mathcal{D}}[f(x) = 1 | x \in \ell]$ and p to denote $\Pr_{\mathcal{D}}[f(x) = 1]$.

Definition 3 *As in [11], the* uncertainty *of a distribution $\mathcal{D}$ is defined to be $U(\mathcal{D}) = 2\sqrt{p(1-p)}$. Let $\mathcal{L}$ be a partition of X into disjoint subsets (so $X = \bigcup_{\ell \in \mathcal{L}} \ell$). The uncertainty of $\mathcal{L}$ under $\mathcal{D}$ is defined to be $U(\mathcal{D}, \mathcal{L}) = \sum_{\ell \in \mathcal{L}} w_\ell u_\ell$, where $u_\ell = U(\mathcal{D}|_\ell) = 2\sqrt{p_\ell(1-p_\ell)}$ is the uncertainty of the conditional distribution $\mathcal{D}|_\ell$ and $w_\ell = \Pr_{\mathcal{D}}[x \in \ell]$ is referred to as the* weight *of leaf ℓ.*

Given any partition $\mathcal{L}$ of X, there is a natural corresponding predictor for the target function f: on each set $\ell \in \mathcal{L}$, we predict 1 iff $p_\ell > \frac{1}{2}$. The error of this predictor under $\mathcal{D}$ is $\sum_\ell w_\ell \min(p_\ell, 1-$

p_ℓ); note that this is at most $\frac{1}{2}U(\mathcal{L}, \mathcal{D})$ since min is less than geometric mean. Thus, the uncertainty of a partition gives an upper bound on the error of the corresponding predictor.

Definition 4 *The* balanced *distribution $\widehat{\mathcal{D}}$ is an equal average of the distributions $\mathcal{D}|_{f^{-1}(1)}$ and $\mathcal{D}|_{f^{-1}(0)}$, i.e. $\widehat{\mathcal{D}}(S) = \frac{1}{2}\Pr_{\mathcal{D}}[x \in S \mid f(x) = 1] + \frac{1}{2}\Pr_{\mathcal{D}}[x \in S \mid f(x) = 0]$.*

Given access to a noise-free oracle $EX(f, \mathcal{D})$, it is easy to simulate the noise-free oracle $EX(f, \widehat{\mathcal{D}})$; this is done by flipping a coin at random to decide whether to choose a positive or negative example. Then wait until one receives such an example[1].

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. (Branching programs go by various names, such as decision graphs and binary decision diagrams, in different communities.) Branching programs were introduced 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.

3.2 The MM Boosting Algorithm

The MM algorithm iteratively constructs a branching program in which each internal node v is labelled with a 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 $p_\ell = \Pr_{x \in \mathcal{D}}[f(x) = 1 | x \text{ reaches } \ell]$. 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 a classifier.

The MM algorithm is given below. The branching program initially consists of a single leaf. The algorithm repeatedly performs two basic operations:

- **Split a leaf (steps 2-3):** The chosen leaf ℓ becomes an internal node which has two new leaves as its children. The label of this new internal node is a hypothesis generated by the weak learning algorithm when run with the oracle $EX(f, \widehat{\mathcal{D}|_\ell})$ (recall that this distribution is obtained by first conditioning on $x \in \ell$ and then balancing that conditional distribution).

- **Merge two leaves (steps 6-7):** The two leaves ℓ_a and ℓ_b chosen for the merge are replaced by a single leaf ℓ. All edges into ℓ_a and ℓ_b are redirected into ℓ.

Intuitively, splitting a leaf should increase the accuracy of our classifier. In the MM 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 have shown that without merges the size of the resulting decision tree may be exponentially large [11]. The leaves to be merged are chosen so as to minimally increase the overall uncertainty of the partition. The condition in Line 7 ensures that we only perform merges whose cumulative uncertainty increase is substantially less than the uncertainty decrease of the most recently performed split, and

[1] 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 we will abort the simulation after some bounded number of draws.

thus we make progress. The final output hypothesis of the MM booster is the final branching program.

The MM Boosting Algorithm:

Input: desired final error level ϵ
access to γ-weak learner A
access to noise-free example oracle $EX(f, \mathcal{D})$

Recall from the definitions: $w_\ell = \Pr_{\mathcal{D}}[x \text{ reaches leaf } \ell]$, $p_\ell = \Pr_{\mathcal{D}}[f(x) = 1 | x \text{ reaches } \ell]$, $u_\ell = 2\sqrt{p_\ell(1-p_\ell)}$, $\mathcal{D}|_\ell$ is the distribution obtained by conditioning on $x \in \ell$, and $\widehat{\mathcal{D}|_\ell}$ is the balanced version of $\mathcal{D}|_\ell$.

Algorithm:

1. Start with the trivial partition $\mathcal{L} = \{X\}$, so the branching program is a single leaf.
2. **Construct candidate splits**: For each leaf $\ell \in \mathcal{L}$, run the weak learning algorithm A on the balanced distribution on this leaf (i.e. oracle $EX(f, \widehat{\mathcal{D}|_\ell})$) to obtain leaves ℓ_0 and ℓ_1.
3. **Choose best split:** Perform the split that reduces the overall uncertainty the most. Let Δ_S be this reduction, so
$$\Delta_S = \max_\ell \{w_\ell u_\ell - w_{\ell_0} u_{\ell_0} - w_{\ell_1} u_{\ell_1}\}.$$
4. Stop if the error of the current branching program $\leq \epsilon$.
5. Set $\Delta_M := 0$.
6. **Consider candidate merges**: Let $\ell_a \neq \ell_b$ be the two leaves which, if merged into one leaf ℓ, would cause the minimum increase in uncertainty. Let z be this minimum value:
$$z := \min_{\ell_a \neq \ell_b} \{w_{\ell_a} u_{\ell_a} + w_{\ell_b} u_{\ell_b} - w_\ell u_\ell\}.$$
7. **Merge if safe:** If $\Delta_M + z < \Delta_S/2$ then
   - Merge leaves ℓ_a, ℓ_b in the branching program.
   - Set $\Delta_M := \Delta_M + z$.
   - Go to step 6.
8. Otherwise, go to step 2.

3.3 Correctness and efficiency of the MM algorithm

We assume in this section that all probabilities are computed exactly by the MM algorithm. In Section 3.4 we show that our analysis still holds if probabilities are estimated by a polynomial amount of sampling. 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 δ probability of failure. This failure probability can be handled with standard techniques as discussed in Section 3.4.

The following lemma corresponds to Lemma 2 in [11].

Lemma 1 *Suppose for distribution $\mathcal{D}$, hypothesis h satisfies $\Pr_{\widehat{\mathcal{D}}}[h(x) \neq f(x)] \leq \frac{1}{2}-\gamma$. Let $\mathcal{L}$ be the partition induced by h, i.e. $\mathcal{L} = \{h^{-1}(0), h^{-1}(1)\}$. Then $U(\mathcal{L},\mathcal{D}) \leq (1-2\gamma^2)U(\mathcal{D})$.*

Proof: Without loss of generality we write

$$\begin{aligned} P_{\mathcal{D}}[f(x)=1] &= p \\ P_{\mathcal{D}}[h(x)=1 \wedge f(x)=1] &= pa \\ P_{\mathcal{D}}[h(x)=0 \wedge f(x)=1] &= p(1-a) \\ P_{\mathcal{D}}[f(x)=0] &= q=(1-p) \\ P_{\mathcal{D}}[h(x)=1 \wedge f(x)=0] &= qb \\ P_{\mathcal{D}}[h(x)=0 \wedge f(x)=0] &= q(1-b) \end{aligned}$$

so the error of h under $\mathcal{D}|_{f(x)=1}$ is $1-a$ and under $\mathcal{D}|_{f(x)=0}$ is b. Since the error under the balanced distribution is at most $\frac{1}{2}-\gamma$, we have $\frac{1-a+b}{2} \leq \frac{1}{2}-\gamma$ and hence $a-b \geq 2\gamma$.

By definition, $U(\mathcal{D}) = 2\sqrt{pq}$ and that

$$\begin{aligned} U(\mathcal{L},\mathcal{D}) &= 2(pa+qb)\sqrt{\frac{paqb}{(pa+qb)^2}} + 2(p(1-a)+q(1-b))\sqrt{\frac{p(1-a)q(1-b)}{(p(1-a)+q(1-b))^2}} \\ &= 2\sqrt{paqb} + 2\sqrt{p(1-a)q(1-b)} \\ &= U(\mathcal{D})\left(\sqrt{ab}+\sqrt{(1-a)(1-b)}\right). \end{aligned}$$

To finish the proof, we observe that

$$\begin{aligned} \sqrt{ab}+\sqrt{(1-a)(1-b)} &= \frac{1}{2}\sqrt{(a+b)^2-(a-b)^2} + \frac{1}{2}\sqrt{(1-a+1-b)^2-(a-b)^2} \\ &\leq \frac{1}{2}\sqrt{(a+b)^2-4\gamma^2} + \frac{1}{2}\sqrt{(2-(a+b))^2-4\gamma^2} \\ &\leq \sqrt{1-4\gamma^2} \\ &\leq 1-2\gamma^2 \end{aligned}$$

where the second inequality uses the concavity of the function $\sqrt{x^2-\tau}$. ■

Lemma 1 implies that as long as the MM branching program does not have too many leaves, each split performed in line 3 gives a substantial decrease in the overall uncertainty:

Lemma 2 *Suppose that the MM branching program's partition $\mathcal{L}$ has L leaves before executing step 3. Then after performing the split in step 3, the new partition $\mathcal{L}'$ satisfies $U(\mathcal{L}',\mathcal{D}) \leq (1-2\gamma^2/L)U(\mathcal{L},\mathcal{D})$.*

Proof: Since $\mathcal{L}$ has L leaves, some leaf ℓ must have $w_\ell u_\ell \geq \frac{1}{L}U(\mathcal{L},\mathcal{D})$. If this leaf were chosen for the split then by Lemma 1 the uncertainty u_ℓ would be multiplied by at most $1-2\gamma^2$, and hence the overall uncertainty $U(\mathcal{L},\mathcal{D})$ would be multiplied by at most $1-2\gamma^2/L$. Since the actual split chosen is the one which reduces overall uncertainty the most, the lemma holds. ■

Now we show that if the branching program has many leaves, there are merges it can perform which do not increase uncertainty by too much.

Lemma 3 *Suppose that the MM branching program has uncertainty $U = U(\mathcal{L}, \mathcal{D})$ and $L \geq \frac{72}{\gamma^2} \log \frac{4}{U\gamma^2}$ leaves. Then there are two leaves ℓ_a and ℓ_b whose merger would cause the uncertainty to increase by at most $\gamma^2 U/L$, i.e. the resulting partition $\mathcal{L}_{a,b}$ would satisfy $U(\mathcal{L}_{a,b}, \mathcal{D}) \leq (1 + \gamma^2/L)U$.*

Proof: We may assume without loss of generality that there are at least $L/2$ leaves ℓ such that $p_\ell \leq \frac{1}{2}$. (The other case, that there are at least $L/2$ leaves ℓ such that $p_\ell \geq \frac{1}{2}$ follows by symmetry.) Consider what would happen if we were to merge two such leaves ℓ_1 and ℓ_2 which have associated weights w_1 and w_2 and uncertainties $u_1 = U(\mathcal{D}|_{\ell_1}) \leq u_2 = U(\mathcal{D}|_{\ell_2})$. It is easily verified that this would give a leaf ℓ with weight $w = w_1 + w_2$ and uncertainty $u = U(\mathcal{D}_\ell)$ satisfying $u_1 \leq u \leq u_2$ (this uses the fact that $p_1, p_2 \leq \frac{1}{2}$). Consequently, the increase in overall uncertainty resulting from such a merge would be

$$wu - w_1 u_1 - w_2 u_2 \leq w_1(u_2 - u_1) = w_1 u_1 (\frac{u_2}{u_1} - 1). \tag{1}$$

Now we imagine putting the uncertainties of these leaves into disjoint buckets. Consider the $L/8$ intervals

$$\left(\left(1 - \frac{\gamma^2}{9}\right)^i, \left(1 - \frac{\gamma^2}{9}\right)^{i-1}\right]$$

for $i = 1, 2, \ldots, L/8$. (These buckets were used explicitly as part of the algorithm in [15] but our presentation uses them only here in the analysis.) Since $(1-x)^{1/x} \leq 1/e$ for $x \in (0, 1]$, we have

$$\left(1 - \frac{\gamma^2}{9}\right)^{L/8} \leq \left(1 - \frac{\gamma^2}{9}\right)^{\frac{9}{\gamma^2} \log \frac{4}{U\gamma^2}} \leq \frac{\gamma^2 U}{4}$$

and hence these buckets cover at least the interval $(\gamma^2 U/4, 1]$.

Suppose first that at least $L/4$ of the $L/2$ leaves with $p_\ell \leq \frac{1}{2}$ have uncertainty $u_\ell \leq \gamma^2 U/4$. If this is the case then there must be some such leaf with with weight $w_\ell \leq 4/L$. By Equation (1), merging this leaf with any other leaf whose uncertainty is at most $\gamma^2 U/4$ results in an increase in uncertainty of at most $w_\ell \gamma^2 U/4 \leq \gamma^2 U/L$, which suffices to establish the lemma in this case.

So now suppose that at least $L/4$ of the $L/2$ leaves with $p_\ell \leq \frac{1}{2}$ have uncertainty $u_\ell > \gamma^2 U/4$. By the pigeon-hole principle, among these $L/4$ values of u_ℓ at least $L/8$ fall into buckets in which they are not the unique largest value assigned to that bucket. Among these $L/8$ values, let ℓ' be the leaf with lowest $w_{\ell'} u_{\ell'}$. Since the total uncertainty is U, we must have $w_{\ell'} u_{\ell'} \leq 8U/L$. Let ℓ'' be a leaf which falls into the same bucket and satisfies

$$u_{\ell'} \leq u_{\ell''} \leq u_{\ell'}/(1 - \gamma^2/9).$$

From Equation (1), the increase in uncertainty which would result from merging ℓ' and ℓ'' is at most

$$\frac{8U}{L}\left(\frac{1}{(1-\gamma^2/9)} - 1\right) = \frac{8U}{L} \cdot \frac{\gamma^2}{9 - \gamma^2} \leq \frac{U\gamma^2}{L}$$

so the lemma is proved. ■

Now we can establish correctness of the MM boosting algorithm:

Theorem 4 *After at most $\frac{144}{\gamma^4} \log \frac{2}{\epsilon\gamma^2} \log \frac{1}{2\epsilon}$ splits and merges, the MM algorithm will output a hypothesis h such that $\Pr_{\mathcal{D}}[h(x) \neq f(x)] \leq \epsilon$.*

Proof: First note that since the algorithm halts as soon as the error $\Pr_{\mathcal{D}}[h(x) \neq f(x)]$ is at most ϵ, throughout its execution we have $U(\mathcal{L}, \mathcal{D}) > 2\epsilon$ (recall that the uncertainty is always at least twice the error rate). We now show that the algorithm halts after the claimed number of steps.

We first note that the number of leaves in the branching program whenever Step 3 is executed is never greater than $L = \frac{72}{\gamma^2} \log \frac{2}{\epsilon\gamma^2}$. To see this, note that if there are L leaves and a split is performed, then by Lemma 2 the uncertainty U prior to the split decreases by at least $2\gamma^2 U/L$. Lemma 3 then implies that there is some merge which would increase the uncertainty by at most $\gamma^2 U/L$. Thus this merge will be performed in Step 7 and there will again be at most L leaves.

Thus by Lemma 2 and the condition in Step 7, the cumulative effect of a split and the (possibly empty) sequence of merges which follows it before the next split is to multiply the uncertainty by at most $(1 - \gamma^2/L)$. Since the uncertainty of the initial trivial partition is at most 1, we have that immediately before the $(s+1)$st split takes place the uncertainty is at most $\left(1 - \frac{\gamma^2}{L}\right)^s \leq e^{-s\gamma^2/L}$. This is less than 2ϵ for $s = \frac{L}{\gamma^2} \log \frac{1}{2\epsilon}$, so at most this many splits take place. The total number of merges is clearly at most the total number of splits, so the theorem is proved. ■

3.4 Approximating MM via sampling

So far we have discussed an idealized version of the MM algorithm in which all probabilities can be computed exactly. In [15] the MM algorithm was run on a fixed sample so this exact computation could in fact be done, but for our extension to the noisy setting it is more convenient to consider a "boosting-by-filtering" version where we do not use a fixed sample. Hence we cannot compute probabilities exactly but instead must use empirical estimates obtained by calling $EX(f, \mathcal{D})$.

Let L be as in Theorem 4. We first note that in Step 2 the algorithm need not run the weak learning algorithm on any leaf ℓ which has $w_\ell u_\ell \leq \frac{\epsilon}{2L}$, since the total contribution of such leaves to the final uncertainty will be at most $\frac{\epsilon}{2}$. By the analysis in Section 3.3, for each leaf ℓ it suffices to estimate the quantity $w_\ell u_\ell$ to additive accuracy $O(\frac{\gamma^2 \epsilon}{L})$. This accuracy ensures that, as in Theorem 4, before the $(s+1)$st split the uncertainty is at most $(1 - \Omega(\gamma^2/L))^s$, and that our final estimate of the uncertainty $\sum_\ell w_\ell u_\ell$ will be off by at most $O(\epsilon)$.

How much time is required to estimate $w_\ell u_\ell$ to a given additive accuracy? We can rewrite $w_\ell u_\ell$ as $2\sqrt{a_\ell b_\ell}$ where $a_\ell = \Pr_{\mathcal{D}}[x \in \ell \text{ and } f(x) = 1]$ and $b_\ell = \Pr_{\mathcal{D}}[x \in \ell \text{ and } f(x) = 0]$. Tail inequalities, such as Chernoff bounds, imply that these probabilities, and hence $w_\ell u_\ell$ as well, can be estimated to any inverse polynomial additive accuracy from a polynomial number of calls to $EX(f, \mathcal{D})$. (Note that from the above discussion, we only need to simulate $EX(f, \widehat{\mathcal{D}|_\ell})$ in Step 2 if $w_\ell u_\ell$ is $\Omega(\epsilon/L)$, and if this is the case then we can simulate each call to $EX(f, \widehat{\mathcal{D}|_\ell})$ in poly(L/ϵ) time with high probability.)

Finally, we note by a standard analysis the total failure probability of all estimates and calls to the weak learner can be bounded by δ at little cost. We thus have:

Theorem 5 *For any $\epsilon, \delta > 0$, if the MM boosting algorithm is run using a γ-weak learner and a noise-free example oracle $EX(f, \mathcal{D})$, then it runs for poly$(\frac{1}{\gamma}, \frac{1}{\epsilon}, \frac{1}{\delta})$ time steps and with probability $1 - \delta$ outputs a hypothesis h satisfying* $\Pr_{\mathcal{D}}[h(x) \neq f(x)] \leq \epsilon$.

4 MMM: Boosting to the noise rate

In this section we modify the MM algorithm to obtain the MMM algorithm which can achieve any accuracy up to the noise rate. The MMM algorithm is given access to a noise-tolerant γ-weak learning algorithm and to a noisy example oracle $EX(f, \mathcal{D}, \eta)$ and is given a value $\tau > 0$; its goal is to output a hypothesis h such that $\Pr_{\mathcal{D}}[h(x) \neq f(x)] \leq \eta + \tau$. We analyze the algorithm in terms of the true probabilities $p_\ell = \Pr_{\mathcal{D}}[f(x) = 1 | x \in \ell]$ instead of the "noisy" probabilities $\tilde{p_\ell} = \Pr_{\mathcal{D}}[\text{label} = 1 | x \in \ell]$. Since $\tilde{p_\ell} = p_\ell(1-\eta) + (1-p_\ell)\eta$, we have

$$p_\ell = \frac{\tilde{p_\ell} - \eta}{1 - 2\eta}. \tag{2}$$

Thus the MMM algorithm can estimate p_ℓ to within an additive error of c by estimating $\tilde{p_\ell}$ to within an additive $\frac{c}{1-2\eta}$. We assume throughout this section that the MMM algorithm knows the value of η. If not, we can use the following standard trick: if we could "guess" η then the algorithm would succeed. In fact, if we could guess η to within a small error, then we would succeed as well. This is because the algorithm would succeed with high probability if the true distribution had our guessed amount of noise, and the two distributions with different amounts of noise are very close (so close that no algorithm that draws a sufficiently small number of examples can succeed on one and fail on the other). Thus, one searches through the possible noise values, starting at small eta and gradually increasing, each time rerunning the algorithm with the estimated η. When we reach the correct value of η, the algorithm will succeed and we will be able to tell by our sufficiently high accuracy.

The MMM algorithm differs from the MM algorithm in the following ways:

- In Step 2 the oracle $EX(f, \widehat{\mathcal{D}|_\ell}, \eta')$, i.e. a noisy balanced oracle, is used to run the weak learning algorithm, where $\eta' > \eta$ is some higher noise rate. (Later we will show how to efficiently simulate $EX(f, \widehat{\mathcal{D}}, \eta')$ given access to $EX(f, \mathcal{D}, \eta)$ and will show that η' is bounded away from $\frac{1}{2}$; this ensures that at each stage the noise-tolerant weak learner can construct a weak hypothesis as required.)

- For $\tau > 0$ define $\mathcal{L}_\tau$ to be the set of leaves ℓ such that $\min\{p_\ell, 1-p_\ell\} \geq \eta + \frac{\tau}{2}$. Each time a leaf ℓ is formed, if $\ell \notin \mathcal{L}_\tau$ then we view ℓ as "dead" and never consider it again for splits or merges; so MMM only performs splits and merges on leaves in $\mathcal{L}_\tau$. (This ensures that we can efficiently simulate the noisy balanced oracle. For leaves not in $\mathcal{L}_\tau$ we may not be able to simulate such an oracle.)

- In Step 4 the algorithm halts if $\Pr_{\mathcal{D}}[h(x) \neq f(x)] \leq \eta + \tau$.

We have the following analogue of Theorem 4:

Theorem 6 *After* $O(\frac{1}{\gamma^4} \log \frac{1}{\tau\gamma} \log \frac{1}{\tau})$ *splits and merges, the MMM algorithm will output a hypothesis h such that* $\Pr_{\mathcal{D}}[h(x) \neq f(x)] \leq \eta + \tau$.

Proof: The error $\Pr_{\mathcal{D}}[h(x) \neq f(x)]$ has contributions from leaves in $\mathcal{L}_\tau$ and not in $\mathcal{L}_\tau$. By definition of $\mathcal{L}_\tau$ the total contribution from leaves not in $\mathcal{L}_\tau$ is at most $\eta + \tau/2$. Thus it suffices to bound the error contribution from leaves in $\mathcal{L}_\tau$ by $\tau/2$. The analysis establishing this bound is very similar to that of Theorem 4 with $\tau/2$ in place of ϵ. Let $U_\tau = \sum_{\ell \in \mathcal{L}_\tau} w_\ell 2\sqrt{p_\ell(1-p_\ell)}$ be the total uncertainty of leaves in $\mathcal{L}_\tau$. As before, it suffices to reduce U_τ to τ. If we set $L_\tau = |\mathcal{L}_\tau|$, then Lemma 2 now holds with $1 - 2\gamma^2/L_\tau$ in place of $1 - 2\gamma^2/L$, because the leaf of largest uncertainty in $\mathcal{L}_\tau$ can be split and its uncertainty reduced by a factor of $1 - 2\gamma^2$. Lemma 3 applies to the subset of leaves $\mathcal{L}_\tau$ and the uncertainty U_τ, so as before if there are many leaves in $\mathcal{L}_\tau$ then merging some pair increases uncertainty by at most $1 + \gamma^2/L_\tau$. Thus, by the same argument as Theorem 4 the value U_τ will be reduced to τ in the same number of splits and merges as in Theorem 4 for $\epsilon = \tau/2$. ■

We now show how to simulate the noisy balanced example oracle $EX(f, \widehat{\mathcal{D}}, \eta')$ using $EX(f, \mathcal{D}, \eta)$. Assume without loss of generality that $p = \Pr_{\mathcal{D}}[f(x) = 1] \leq \frac{1}{2}$. From the discussion above we may assume that $p \geq \eta + \frac{\tau}{2}$. We filter examples from $EX(f, \mathcal{D}, \eta)$ as follows:

- **Labeled 0:** Reject each example labeled 0 with probability $\frac{1-2p}{1-p-\eta}$, otherwise keep it.
- **Labeled 1:** Flip to 0 with probability $\frac{(1-2p)\eta(1-\eta)}{(1-\eta-p)(p+\eta-2p\eta)}$, otherwise don't flip the label.

The idea is that the rejection balances the distribution between true positive and true negative examples, but as a result of this balancing we now have asymmetric noise, i.e. the fraction of negative examples that are mislabelled is greater than the fraction of positive examples that are mislabelled. To compensate, the flipping causes an equal fraction of positive and negative examples to be mislabelled, so we have true classification noise at a higher rate η'. We have the following lemma:

Lemma 7 *Given access to $EX(f, \mathcal{D}, \eta)$, where $p = \Pr[f(x) = 1]$ and $\min\{p, 1-p\} \geq \eta + \frac{\tau}{2}$, by making poly$(\frac{1}{\tau}, \log \frac{1}{\delta})$ calls to $EX(f, \mathcal{D}, \eta)$ we can simulate a call to $EX(f, \widehat{\mathcal{D}}, \eta')$ with probability $1 - \delta$, where $\eta' \leq \frac{1}{2} - \frac{\tau}{4}$.*

Proof: Recall that we have access to a noisy example oracle $EX(f, \mathcal{D}, \eta)$ where $\mathcal{D}$ is some distribution, $0 < \eta < \frac{1}{2}$ is the noise rate, and $p = \Pr_{\mathcal{D}}[f(x) = 1]$ satisfies $\eta + \frac{\tau}{2} \leq p \leq \frac{1}{2}$ for some $\tau > 0$. We show how this oracle can be used to simulate the oracle $EX(f, \widehat{\mathcal{D}}, \eta')$. Here $\widehat{\mathcal{D}}$ is the balanced version of distribution $\mathcal{D}$ and $0 < \eta' < \frac{1}{2}$ is a new noise rate.

We filter examples from $EX(f, \mathcal{D}, \eta)$. For each example,

Labeled 0: Reject with probability $p_r = \frac{1-2p}{1-p-\eta}$, keep with probability $1 - p_r = \frac{p-\eta}{1-p-\eta}$.
Labeled 1: Flip its label with probability $p_f = \frac{(1-2p)\eta(1-\eta)}{(1-p-\eta)(p+\eta-2p\eta)}$, don't flip with probability $1 - p_f$.

We will show that this results in $EX(f, \widehat{\mathcal{D}}, \eta')$ where $\eta' \leq \frac{1}{2} - \frac{\tau}{4}$.

In order to verify this, it suffices to check the following two things. First, with regard to rejection,

$$\Pr_{\mathcal{D}}[f(x) = 0 \wedge \text{not rejected}] = \Pr_{\mathcal{D}}[f(x) = 1 \wedge \text{not rejected}].$$

This would show that at least the resulting distribution is balanced but says nothing about the labels or apparent noise rates. The LHS above can be written as $(1-p)((1-\eta)(1-p_r)+\eta)$ because the example was negative with probability $1-p$ and either the example was not noisy (probability $1-\eta$), thus labeled 0, and kept (probability $1-p_r$), or it was noisy (probability η) and was kept for sure. Similarly, the RHS above can be written as $p(\eta(1-p_r)+1-\eta)$. One can check that the above two quantities are both $\frac{(1-2\eta)p(1-p)}{1-p-\eta}$.

Second, we need to check that the noise rates on both positive and negative examples are η'. In other words, we need to verify that,

$$\Pr_{\mathcal{D}}[f(x) = 0 \wedge \text{not rejected} \wedge \text{label}' = 1] = \eta' \Pr_{\mathcal{D}}[f(x) = 0 \wedge \text{not rejected}]$$

and

$$\Pr_{\mathcal{D}}[f(x) = 1 \wedge \text{not rejected} \wedge \text{label}' = 0] = \eta' \Pr_{\mathcal{D}}[f(x) = 1 \wedge \text{not rejected}].$$

In the above, label$'$ is the possibly flipped label after step 2. The first LHS can be written as $(1-p)\eta(1-p_f)$ because the example must have been a negative example that was noisy and not flipped. Similarly, the second LHS above is $p(\eta(1-p_r)+(1-\eta)p_f)$. A tedious but straightforward verification shows that these two quantities are both $\frac{\eta(1-p)}{(p+\eta-2p\eta)} \cdot \frac{(1-2\eta)p(1-p)}{(1-p-\eta)}$.

Based on our earlier calculation that

$$\Pr[f(x) = 0 \wedge \text{not rejected}] = \frac{(1-2\eta)p(1-p)}{1-p-\eta},$$

the effective noise rate is

$$\eta' = \frac{\eta(1-p)}{p+\eta-2p\eta} = \frac{1}{2} - \frac{p-\eta}{2(p+\eta-2p\eta)}.$$

It is straightforward to verify that $\eta' \leq \frac{1}{2} - \frac{\tau}{4}$ because $p - \eta \geq \frac{\tau}{2}$ and $p+\eta-2p\eta < 1$, so the lemma is proved. $\blacksquare$

As in Section 3.4, to run MMM successfully we need only estimate each w_ℓ, p_ℓ, u_ℓ to inverse polynomial accuracy. A new issue which arises is that since p_ℓ is an estimate instead of a precise value, the filtering procedure described above to sample from $EX(f, \widehat{\mathcal{D}|_\ell}, \eta')$ will not perfectly simulate this oracle, i.e. the resulting distribution may not be perfectly balanced, and the noise rates on positive and negative examples may not be exactly equal. However, this is not a problem since a straightforward analysis shows that the statistical difference between the true distribution and the distribution we simulate can be made as small as any inverse polynomial (at the expense of a corresponding polynomial increase in runtime). Thus any weak learner which makes polynomially many draws from our simulated distribution cannot distinguish between it and the true distribution with high probability. Since it succeeds with high probability from the true distribution, it must succeed with high probability from the simulated distribution as well.

We thus have:

Theorem 8 *For any $\tau, \delta > 0$, if the MMM boosting algorithm is run using a noise-tolerant γ-weak learner and a noisy source of examples, $EX(f, \mathcal{D}, \eta)$, then it runs for poly$(\frac{1}{\gamma}, \frac{1}{\tau}, \frac{1}{\delta}, \frac{1}{1-2\eta})$ time steps and with probability $1-\delta$ outputs a hypothesis h satisfying* $\Pr_{\mathcal{D}}[h(x) \neq f(x)] \leq \eta + \tau$.

In the next section, we give a lower bound showing that, in general, it is impossible to boost a black-box weak learner past the noise rate.

5 Boosting past the noise rate is hard

The basic approach here is that we suppose we have some distribution with a $p < \eta$ fraction of positive examples. Thus the all 0's hypothesis is a good weak hypothesis to start. We will describe an "unboostable" weak learner with the following property: whenever possible, it outputs a trivial hypothesis that contains no useful information. In fact, the weak learner only does something interesting if its sample contains a large set of unique (occuring only once in the sample) examples that is nearly 1/2 positive. The motivation for considering this weak learner is that it is difficult for a booster to generate a set of examples that is nearly 1/2 positive, because a random example that is labeled positive is still more than 1/2 likely to be a true negative example, and thus intuitively it is hard for the booster to make the weak learner give any useful information.

However, there is a difficulty in that the booster might conceivably be able to learn on its own, without even using the weak learner. Thus, in order to prove that it is hard to boost past the noise rate, we somehow need to ensure that the booster must indeed use the weak learner.

Our approach takes advantage of the fact that since a boosting algorithm must work for any concept class, the booster does not "know" the concept class on which it is being run.[2] We will consider concept classes each containing a single function; for each such concept class there is a corresponding weak learner which knows this function (since the weak learner may be tailored for the particular concept class being learned), but the booster does not. The overall collection of functions (collection of concept classes) considered will be a pseudo-random family of functions, so intuitively the booster should be unable to learn without using the weak learner.

Using this approach, we prove the following:

Theorem 9 *If one-way functions exist then black-box noise-tolerant boosters do not exist.*

In fact we show (Theorem 13) that for any $\tau > 0$ it is cryptographically hard to boost to accuracy $\eta - \tau$ in the presence of classification noise at rate η.

We give some more intuition for our construction. The unboostable weak learning algorithm is as follows. Consider a target function f which has only an $\eta - \tau$ fraction of inputs x satisfying $f(x) = 1$. Then under the uniform distribution a weak learner can output the constant-0 hypothesis; in fact the only distributions for which a weak learner must output some other hypothesis are nonuniform ones which put weight at least 1/2 on the small set of positive examples. Thus the only way a boosting algorithm can get anything useful out of such a weak learner is to simulate a distribution which puts weight at least 1/2 on positive examples, and as argued earlier this seems difficult to do since the noise rate is η.

In fact there is a hole in this argument. For example, a boosting algorithm could simulate a distribution which puts weight 1/2 on each of two examples. If the booster is lucky and one of the examples is positive, then the resulting distribution is balanced. Thus, in order to design a maximally unhelpful weak learner which thwarts this boosting strategy, we have our weak learner make a lookup table of examples which it sees many times in its sample. For each example in the table, the weak learner's output is the majority vote label from its occurrences in the sample; on all other examples the weak learner outputs 0. Intuitively, this hypothesis is sufficient to satisfy the weak learning criterion unless the data set for the weak learner contains a large number of distinct instances many of which are true positive examples; only if this is the case does the weak learner give up some useful information.

Now we give the actual construction. Let $0 < p < 1$. Let $\{f_s : \{0,1\}^{|s|} \rightarrow \{0,1\}\}_{s \in \{0,1\}^*}$ be a p-biased pseudorandom function family, i.e. a family of functions which are indistinguishable from truly random p-biased functions (see Appendix A for a formal definition of p-biased pseudorandom function family). For each $s \in \{0,1\}^n$ we define a concept class C_s as follows: each class C_s contains exactly one concept, which is f_s.

[2] An alternative approach would instead be to assume that the boosting algorithm cannot use any information about the particulars of the learning problem. Namely, we could assume that the boosting algorithm cannot do anything with examples other than identify whether two are the same or different, examine their labels, and apply the weak hypotheses to them. Under this assumption almost any concept class can be shown to have an unboostable weak learner. In our cryptographic construction described below, we bypass this strong assumption by instead assuming that one-way functions exist.

Fix $0 < \gamma < \frac{1}{4}$. We now define an algorithm A_s for each concept class C_s. In the following description the values m_1, k, m_2, are polynomials in $n, \frac{1}{\gamma}, \frac{1}{1-2\eta}, \frac{1}{\delta}$ whose values will be given later.

Algorithm $A_s(\eta, \gamma)$:

1. Draw a sequence S_1 of m_1 examples. (Note that a given instance $x \in \{0,1\}^n$ may occur more than once in S_1.)
2. Let T be the set of instances $x \in \{0,1\}^n$ which occur at least k times in S_1. For each $x \in T$ let $b_x \in \{0,1\}$ be the majority vote label of all pairs $\langle x, y \rangle$ in S_1 which have x as the instance.
3. Define h_1 to be the hypothesis $h_1(x) \equiv$ "if $x \in T$ then output b_x else output 0."
4. Draw a sequence S_2 of m_2 examples. Abort and output the hypothesis h_1 if there is any instance x which occurs more than once in S_2 but is not in T.
5. Let N be the number of occurrences in S_2 of instances x such that $x \notin T$ and $f_s(x) = 1$. If $N \geq (\frac{1}{2} - \frac{3\gamma}{2})m_2$ then output f_s, and otherwise output h_1.

Note that the hypothesis h_1 is quite uninformative since any algorithm with access to the example oracle can generate this hypothesis for itself without using A_s. Steps 4 and 5 ensure that the informative f_s hypothesis is output only if S_2 contains many distinct positive examples.

The following claim shows that A_s is indeed a noise-tolerant weak learning algorithm. As before, we assume that we know the noise rate, but again this assumption can be removed.

Claim 10 *A_s is a noise-tolerant γ-weak learning algorithm for concept class C_s.*

Proof: The values m_2, m_1 and k are polynomials in $n, \frac{1}{\gamma}, \frac{1}{1-2\eta}, \frac{1}{\delta}$ which will be defined later.

We first observe that A_s runs in polynomial time. To see this, note that A_s can have f_s "hard-wired" into it, and f_s is efficiently evaluable, so the number N in Step 5 can be computed exactly in polynomial time.

It remains to show that for any distribution $\mathcal{D}$ and any $0 < \eta < \frac{1}{2}$, if A_s is run using $EX(f_s, \mathcal{D}, \eta)$ as the oracle, then with probability at least $1 - \delta$, A_s outputs a hypothesis h such that $\Pr_{\mathcal{D}}[h(x) \neq f_s(x)] \leq \frac{1}{2} - \gamma$. We use the following two lemmas which we prove later:

Lemma 11 *A_s aborts in line 4 with probability less than $\frac{\delta}{3}$.*

Lemma 12 *With probability at least $1 - \frac{\delta}{3}$, we have $b_x = f_s(x)$ for every $x \in T$.*

We will analyze an alternate algorithm A'_s in which the test in line 4 is not performed and b_x is defined to equal $f_s(x)$ for every $x \in T$. By Lemmas 11 and 12 it suffices to show that $\Pr_{\mathcal{D}}[h'(x) \neq f(x)] \leq \frac{1}{2} - \gamma$ with probability at least $1 - \frac{\delta}{3}$, where h' is the hypothesis output by A'_s. Consequently, it suffices to show that if $\Pr_{\mathcal{D}}[h'_1(x) \neq f(x)] > \frac{1}{2} - \gamma$ then A'_s outputs f_s with probability $1 - \frac{\delta}{3}$.

To see that this condition holds, note that in line 5 of A'_s we have that S_2 is a set of independent random draws from $EX(f_s, \mathcal{D}, \eta)$. (This is not true in line 5 of A_s since in A_s we have conditioned on S containing no repeated instances which are not in T.) Thus in A'_s the value N is an empirical estimate of $\Pr_{\mathcal{D}}[x \notin T \text{ and } f_s(x) = 1]$ obtained from m_2 independent samples. As long as $m_2 \geq 2(\log \frac{3}{\delta})/\gamma^2$, standard Chernoff bounds tell us that with probability at least $1 - \frac{\delta}{3}$ the fraction N/m_2 differs from $\Pr_{\mathcal{D}}[x \notin T \text{ and } f_s(x) = 1]$ by at most $\frac{\gamma}{2}$. Hence if $\Pr_{\mathcal{D}}[x \notin T \text{ and } f_s(x) = 1]$ is greater than $\frac{1}{2} - \gamma$ we output f_s with probability at least $1 - \frac{\delta}{3}$. Since in A'_s hypothesis h'_1 is guaranteed to be right on $x \in T$, we have $\Pr_{\mathcal{D}}[h'_1(x) \neq f(x)] = \Pr_{\mathcal{D}}[x \notin T$ and $f(x) = 1]$ and the claim is proved. (Claim 10) ■

Proof of Lemma 11: For $1 \leq i < j \leq m_2$, call positions (i, j) in S_2 a *violator* if the corresponding elements are equal, i.e. $x_i = x_j$, and the number of occurrences of x_i in S_1 is less than k. The algorithm aborts in Step 4 only if there is some violator (i, j). We now upper bound the probability that any particular (i, j) is a violator.

Fix (i, j) and also fix x_i. We may imagine that S_1 and x_j were drawn in the following way: First a multiset S' of $m_1 + 1$ labeled examples was drawn from the example oracle, and then a random element of S' was chosen to be x_j and the rest were chosen for S_1. This is equivalent to drawing x_j and all examples in S_1 independently from the example oracle.

Now suppose that there were t occurrences of x_i in S'. If $t > k$, then there is no way that (i, j) can be a violator because there will always be at least k occurrences of x_i in S_1. On the other hand, the probability that $x_j = x_i$ is exactly $t/(m_1 + 1)$. So if $t \leq k$, the probability that (i, j) is a violator is $t/(m_1 + 1) < k/m_1$.

By the union bound, the probability that any (i, j) is a violator is at most $m_2^2 k/m_1$. This is at most $\delta/3$ provided that $m_1 \geq 3m_2^2 k/\delta$. (Lemma 11) ■

Proof of Lemma 12: Fix any $x \in T$, so x occurs $m \geq k$ times in S_1. The probability that the majority vote of the labels corresponding to instances of x in S_1 is incorrect is precisely the probability that a coin which has probability $\eta < \frac{1}{2}$ of coming up HEADS comes up HEADS more often than TAILS in $m \geq k$ tosses. Using a standard Chernoff bound, as long as $k \geq 2(\log \frac{3m_1}{\delta})/(1 - 2\eta)^2$ this probability is at most $\frac{\delta}{3m_1}$, so the probability that $b_x \neq f_s(x)$ for any fixed $x \in T$ is at most $\frac{\delta}{3m_1}$. Since T contains at most m_1 instances, a union bound finishes the proof. (Lemma 12) ■

So we have seen that the above three lemmas hold as long as $m_2 \geq 2(\log \frac{3}{\delta})/\gamma^2$, $m_1 \geq 3m_2^2 k/\delta$, and $k \geq 2(\log \frac{3m_1}{\delta})/(1 - 2\eta)^2$, which is easily achieved for polynomial sized m_1, m_2, and k.

5.1 Proof of Theorem 9

Let $\mathcal{U}$ denote the uniform distribution on $\{0, 1\}^n$. Fix any noise rate $0 < \eta < \frac{1}{2}$ and any $0 < \tau < \eta$. Fix $p = \eta - \frac{\tau}{2}$. Let the parameter in algorithm A_s be $\gamma = \frac{\eta - p}{4(\eta + p - 2\eta p)} < \frac{1}{4}$. We prove Theorem 9 by establishing the following stronger theorem, which bounds the accuracy level that black-box boosting algorithms can achieve in the presence of noise at rate η.

Theorem 13 *Let $\{f_s\}$ be a p-biased pseudorandom function family. Then, for random s, no black-box boosting algorithm B, given access to $EX(f_s, \mathcal{U}, \eta)$ and A_s, can output a hypothesis whose error is at most $\eta - \tau$. More precisely: for all polynomials Q and all polynomial time* *algorithms B, for n sufficiently large,*

$$\Pr_{s\in\mathcal{U}}[\Pr_{x\in\mathcal{U}}[h(x) \neq f_s(x)] \leq \eta - \tau] < \frac{1}{Q(n)}$$

where h is the hypothesis output by B.

Theorem 13 gives a lower bound of η on the accuracy level ϵ which any polynomial time black box boosting algorithm can achieve. In Section 4 we analyzed the MMM boosting algorithm (which is black-box) and showed that it matches this lower bound: given any $\epsilon = \eta + \tau$ where $\tau > 0$, the MMM algorithm achieves ϵ-accuracy in the presence of classification noise at rate η in time polynomial in $\frac{1}{\tau}$ (and the other relevant parameters). Thus the bound of Theorem 13 (and of the MMM algorithm) is the best possible.

The idea of the proof of Theorem 13 is that B will only succeed if A_s outputs f_s at some invocation. As above, this can only happen if S_2 contains at least a $(\frac{1}{2} - \frac{3\gamma}{2})$ fraction of distinct positive examples. Since f_s is a p-biased pseudorandom function and the noise rate η is sufficiently larger than p, such a set S_2 is difficult to construct.

Before giving the proof we introduce some terminology: we say that the set S_2 is *foolproof* if $N \geq (\frac{1}{2} - \frac{3\gamma}{2})m_2$ and otherwise we say that S_2 is *foolable.* We write $B^{\mathcal{O},A}$ to indicate that B has access to the example oracle $\mathcal{O}$ and black-box access to the weak learning algorithm A. We say that $B^{\mathcal{O},A_s}$ *hits* f_s if at some point during its execution B invokes A_s and A_s draws a foolproof sequence S_2 in Step 4 (so if A_s does not abort in Step 4, it outputs hypothesis f_s in Step 5). We say that it *misses* if it does not hit. We say that a hypothesis h is *good* if $\Pr_{x\in\mathcal{U}}[h(x) \neq f_s(x)] \leq \eta - \tau$.

Theorem 13 follows immediately from the following two lemmas. Here and subsequently we write "p.p.t." as an abbreviation for "probabilistic polynomial time."

Lemma 14 *For all polynomials Q, all p.p.t. algorithms B, and all sufficiently large n,*

$$\Pr[B^{EX(f_s,\mathcal{U},\eta),A_s} \text{ hits } f_s] < \frac{1}{Q(n)}.$$

Lemma 15 *For all polynomials Q, all p.p.t. algorithms B, and all sufficiently large n,*

$$\Pr[B^{EX(f_s,\mathcal{U},\eta),A_s} \text{ outputs a good } h \mid B^{EX(f_s,\mathcal{U},\eta),A_s} \text{ misses } f_s] < \frac{1}{Q(n)}.$$

5.2 Proof of Lemma 14

The idea of the proof is as follows: before hitting f_s for the first time, algorithm A_s outputs the hypothesis h_1 from Step 4 each time it is invoked by B. However, it is not difficult to see that B can generate this hypothesis for itself without having any access to A_s. Thus, prior to its first call of A_s which hits f_s, B might as well have access only to $EX(f_s,\mathcal{U},\eta)$. We then show that no p.p.t. algorithm which has access only to $EX(f_s,\mathcal{U},\eta)$ can hit f_s with nonnegligible probability. Intuitively, the reason why B cannot do this is because the frequency of positive examples is low relative to the noise rate η, so even examples $\langle x, 1\rangle$ from $EX(f_s,\mathcal{U},\eta)$ have too low a probability of being true positive examples to be useful.

More formally, let B be any p.p.t. algorithm. We may assume that for all oracles $\mathcal{O}$ and algorithms A, the algorithm $B^{\mathcal{O},A}$ makes exactly q queries to $\mathcal{O}$ and exactly t calls to A, where q, t are both poly(n). For $i = 1, \ldots, t$ let X^i denote the sequence $x^{i,1}, \ldots, x^{i,|S_2|}$ of strings which

$B^{EX(f_s,\mathcal{U},\eta),A_s}$ provides to algorithm A_s in Step 3 of the ith invocation of A_s. Each X^i is thus a random variable over the probability space defined by the uniform choice of $s \in \{0,1\}^n$ and any internal randomness of algorithm B. For succinctness, we say that X^i *hits* f_s if X^i is foolproof and does not cause A_s to abort in Step 3.

For each $s \in \{0,1\}^n$ let $\tilde{A}_s$ be a modified version of algorithm A_s which always outputs h_1. Consider the following algorithm $\tilde{B}$ which takes access only to $EX(f_s,\mathcal{U},\eta)$:

- Algorithm $\tilde{B}^{EX(f_s,\mathcal{U},\eta)}$ first simulates the execution of $B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s}$ (note that $\tilde{B}$ can simulate $\tilde{A}_s$ for itself given access to $EX(f_s,\mathcal{U},\eta)$).
- Algorithm $\tilde{B}$ then chooses a uniform random value $1 \leq \ell \leq t$ and outputs the sequence $\tilde{X}^i$ of strings $\tilde{x}^{\ell,1},\dots,\tilde{x}^{\ell,|S_2|}$ which $B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s}$ provided to algorithm $\tilde{A}_s$ in Step 3 of the ℓth invocation of $\tilde{A}_s$.

Now, without loss of generality, we may assume that $\tilde{X}^i = X^i$ for all i (i.e. the random variables $\tilde{X}^i$ and X^i are identically distributed for all i). To see this, note that at each invocation A_s outputs either h_1 or f_s; the $\tilde{X}^i$'s correspond to having A_s always output h_1. But even if A_s outputs f_s at some call, we may assume without loss of generality that the boosting algorithm B stores f_s but continues running just as if A_s outputted h_1 (recall that the booster can construct such a h_1 for itself using $EX(f_s,\mathcal{U},\eta)$). For such a booster, each $\tilde{X}^i$ will be identical to the corresponding X^i.

We thus have that

$$\Pr[\tilde{X}^\ell \text{ hits } f_s] = \Pr[X^\ell \text{ hits } f_s] \geq \Pr[B^{EX(f_s,\mathcal{U},\eta),A_s} \text{ hits } f_s]/t.$$

This, together with the following lemma, implies Lemma 14.

Lemma 16 $\Pr[\tilde{X}^\ell$ *hits* $f_s] < \frac{1}{Q(n)}$ *for all polynomials* Q *and all sufficiently large* n.

Proof of Lemma 16: Let f be a Boolean function from $\{0,1\}^n$ to $\{0,1\}$. Consider the following algorithm D which takes access to an oracle for f and outputs a single bit:

- D^f first simulates the execution of $\tilde{B}^{EX(f,\mathcal{U},\eta)}$. D^f simulates each call to $EX(f,\mathcal{U},\eta)$ by choosing a uniform random $x \in \{0,1\}^n$, calling f to obtain $f(x)$, and flipping this bit with probability η. Let $\tilde{Y}^i$ denote the sequence of strings which $B^{EX(f,\mathcal{U},\eta),\tilde{A}_s}$ (which is simulated by $\tilde{B}^{EX(f,\mathcal{U},\eta)}$) provided to algorithm $\tilde{A}_s$ in Step 3 of its ith invocation of $\tilde{A}_s$.
- Let $1 \leq \ell \leq t$ be the value selected by $\tilde{B}^{EX(f,\mathcal{U},\eta)}$. If $\tilde{Y}^\ell$ hits f (meaning that there are at least $N \geq (\frac{1}{2} - \frac{3\gamma}{2})m_2$ uniquely occurring instances in $\tilde{Y}^\ell$ such that $f(x) = 1$), then D^f outputs 1. Otherwise it outputs 0.

Looking over the algorithm, one sees that D is a p.p.t. algorithm. The following claim plays a crucial role in our argument. (Appendix A defines $\mathcal{F}_{n,p}$, a p-biased pseudorandom family of functions.)

Claim 17 *Suppose that* f *is a random function drawn from* $\mathcal{F}_{n,p}$. *Then for all polynomials* Q *and all sufficiently large* n, *we have* $\Pr[D^f$ *outputs* $1] < 1/Q(n)$.

Proof: In order for $\tilde{Y}^\ell$ to hit f, algorithm $\tilde{B}$ must construct a sequence of m_2 instances in $\{0,1\}^n$ which contains $N \geq (\frac{1}{2} - \frac{3\gamma}{2})m_2$ distinct instances with $f(x) = 1$. Since $\tilde{B}$ makes at most polynomially many calls to $EX(f,\mathcal{U},\eta)$, we have that with probability exponentially close to 1, $\tilde{B}$ never receives the same instance more than once from $EX(f,\mathcal{U},\eta)$. Thus we may assume that after it has made all its oracle calls to f, there are three types of instances $x \in \{0,1\}^n$ for $\tilde{B}$:

- Instances x such that $\tilde{B}$ received $\langle x, 1\rangle$ from a call of $EX(f,\mathcal{U},\eta)$. For such an x, either $f(x) = 1$ and the label was not flipped by D or $f(x) = 0$ and the label was flipped by D. Hence for such an x we have that $f(x) = 1$ with probability $\frac{p(1-\eta)}{p(1-\eta)+(1-p)\eta}$.
- Instances x such that $\tilde{B}$ received $\langle x, 0\rangle$ from a call of $EX(f,\mathcal{U},\eta)$. For such an x, either $f(x) = 0$ and the label was not flipped by D or $f(x) = 1$ and the label was flipped by D. Hence for such an x we have that $f(x) = 1$ with probability $\frac{p\eta}{(1-p)(1-\eta)+p\eta}$.
- Instances x such that $\tilde{B}$ never received an example $\langle x, b\rangle$. In this case we have that $f(x) = 1$ with probability p.

We will use the following fact:

Fact 18 $\max\{\frac{p(1-\eta)}{p(1-\eta)+(1-p)\eta}, \frac{p\eta}{(1-p)(1-\eta)+p\eta}, p\} = \frac{1}{2} - 2\gamma$.

Proof: Recall that $0 < \gamma = \frac{\eta-p}{4(\eta+p-2\eta p)} < \frac{1}{4}$, $0 < \eta < \frac{1}{2}$, and $p = \eta - \frac{\tau}{2}$. We thus have $p = \frac{\eta(1-4\gamma)}{1+4\gamma(1-2\eta)}$.

We first show that $p < \frac{1}{2} - 2\gamma$. Substituting for p, multiplying both sides by 2 and rearranging, this inequality becomes $(1-16\gamma^2)(1-2\eta) > 0$ which is clearly true.

Now we show that $\frac{p(1-\eta)}{p(1-\eta)+(1-p)\eta} = \frac{1}{2} - 2\gamma$. This follows from substituting for p and simplifying the left side.

Finally we show that $\frac{p\eta}{(1-p)(1-\eta)+p\eta} < \frac{1}{2} - 2\gamma$. Substituting for p, multiplying both sides by 2 and rearranging, this inequality becomes $\frac{(1-16\gamma^2)(1-2\eta)}{1+4\gamma-8\gamma\eta} > 0$ which is clearly true. (Fact 18) ■

Thus, regardless of how $\tilde{B}$ selects instances of these three types for the sequence of length m_2, the probability that there are at least $(\frac{1}{2} - \frac{3\gamma}{2})m_2$ distinct instances with $f(x) = 1$ is at most the probability that a $(\frac{1}{2} - 2\gamma)$-biased coin comes up HEADS at least $(\frac{1}{2} - \frac{3\gamma}{2})m_2$ times in m_2 flips. As long as $m_2 = \Omega(n/\gamma^2)$, standard Chernoff bounds guarantee this probability to be $1/2^{\Omega(n)}$, and the claim is proved. (Claim 17) ■

By the definition of p-biased pseudorandomness and Claim 17, we have that if f is a p-biased pseudorandom function f_s where s is uniformly chosen in $\{0,1\}^n$, then for all polynomials Q and all sufficiently large n we have $\Pr[D^{f_s}$ outputs 1$] < 1/Q(n)$ as well. However, it is straightforward to verify from the construction of algorithm D that $\Pr[D^{f_s}$ outputs 1$]$ is precisely $\Pr[\tilde{X}^\ell$ hits $f_s]$. This proves the lemma. (Lemma 16) ■

5.3 Proof of Lemma 15

The intuition here is that by conditioning on the event that $B^{EX(f_s,\mathcal{U},\eta),A_s}$ misses f_s, B might as well have access only to $EX(f_s,\mathcal{U},\eta)$. Since f_s is a p-biased pseudorandom function, though, no p.p.t. algorithm can output a good hypothesis (i.e. learn f_s to high accuracy), since otherwise it would be possible for a p.p.t. algorithm to learn a random function from $\mathcal{F}_{n,p}$ to high accuracy which is absurd.

More formally, let B be any p.p.t. algorithm. Consider the following algorithm $\tilde{C}$ which takes access only to $EX(f_s,\mathcal{U},\eta)$: algorithm $\tilde{C}^{EX(f_s,\mathcal{U},\eta)}$ simulates the execution of $B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s}$ and outputs the hypothesis h which $B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s}$ outputs. (Note that $\tilde{C}$ can simulate $\tilde{A}_s$ for itself given access to $EX(f_s,\mathcal{U},\eta)$.)

The following two lemmas together imply Lemma 15:

Lemma 19 *For all sufficiently large n, we have*

$\Pr[\tilde{C}^{EX(f_s,\mathcal{U},\eta)}$ *outputs a good h*$] > \Pr[B^{EX(f_s,\mathcal{U},\eta),A_s}$ *outputs a good h* $\mid B^{EX(f_s,\mathcal{U},\eta),A_s}$ *misses* $f_s]/2$.

Lemma 20 $\Pr[\tilde{C}^{EX(f_s,\mathcal{U},\eta)}$ *outputs a good h*$] < \frac{1}{Q(n)}$ *for all polynomials Q and all large enough n.*

Proof of Lemma 19: We have

$$\begin{aligned}
\Pr[\tilde{C}^{EX(f_s,\mathcal{U},\eta)} \text{ outputs a good } h] &= \Pr[B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ outputs a good } h] \\
&\geq \Pr[B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ outputs a good } h \;\&\; B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ misses } f_s] \\
&= \Pr[B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ outputs a good } h \mid B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ misses } f_s] \cdot \\
&\quad\; \Pr[B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ misses } f_s] \\
&> \Pr[B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ outputs a good } h \mid B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ misses } f_s]/2
\end{aligned}$$

where the last inequality holds for all sufficiently large n by Lemma 14. (Recall that $B^{EX(f_s,\mathcal{U},\eta),A_s}$ can simulate $B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s}$, so we have $\Pr[B^{EX(f_s,\mathcal{U},\eta),A_s}$ misses $f_s] \leq \Pr[B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s}$ misses $f_s]$.) Let $TRANS(B^{EX(f_s,\mathcal{U},\eta),A_s})$ ($TRANS(B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s})$ respectively) denote a complete transcript of algorithm B's execution using $EX(f_s,\mathcal{U},\eta)$ and weak learning algorithm A_S ($\tilde{A}_s$ respectively). $TRANS(B^{EX(f_s,\mathcal{U},\eta),A_s})$ and $TRANS(B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s})$ are both random variables over the probability space defined by choosing s, making random draws to $EX(f_s,\mathcal{U},\eta)$, and any internal randomness of B. Induction shows that the two conditional random variables

$$TRANS(B^{EX(f_s,\mathcal{U},\eta),A_S}) \;|(\; B^{EX(f_s,\mathcal{U},\eta),A_s} \text{ misses } f_s)$$

and

$$TRANS(B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_S}) \;|(\; B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ misses } f_s)$$

are identically distributed. This implies that

$$\begin{aligned}
&\Pr[B^{EX(f_s,\mathcal{U},\eta),A_s} \text{ outputs a good } h \mid B^{EX(f_s,\mathcal{U},\eta),A_s} \text{ misses } f_s] = \\
&\qquad \Pr[B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ outputs a good } h \mid B^{EX(f_s,\mathcal{U},\eta),\tilde{A}_s} \text{ misses } f_s]
\end{aligned}$$

which combined with the previous inequality proves the lemma. ■

Proof of Lemma 20: Let f be a Boolean function from $\{0,1\}^n$ to $\{0,1\}$. Consider the following algorithm E which takes access to an oracle for f and outputs a single bit:

- E^f first simulates the execution of $\tilde{C}^{EX(f,\mathcal{U},\eta)}$. Like algorithm D^f in the previous subsection, E^f simulates each call to $EX(f,\mathcal{U},\eta)$ by choosing a uniform random $x \in \{0,1\}^n$, calling f to obtain $f(x)$, and flipping this bit with probability η. Let h_f be the hypothesis which $\tilde{C}^{EX(f,\mathcal{U},\eta)}$ outputs.
- E^f then selects n independent uniform random n-bit strings $z^1,\ldots,z^n \in \{0,1\}^n$. E^f computes μ which is the fraction of these strings which have $h_f(z^i) = f(z^i)$. E^f outputs 0 if $\mu < 1 - \frac{p+\eta-\tau}{2}$ and outputs 1 if $\mu \geq 1 - \frac{p+\eta-\tau}{2}$.

It is not difficult to see that E is a p.p.t. algorithm. We have:

Claim 21 *Suppose that f is a random function drawn from $\mathcal{F}_{n,p}$. Then for all polynomials Q and all sufficiently large n we have* $\Pr[E^f$ *outputs* $1] < 1/Q(n)$.

Proof of Claim 21: Since $\tilde{C}^{EX(f,\mathcal{U},\eta)}$ makes at most poly(n) many calls to $EX(f,\mathcal{U},\eta)$, with probability $1-1/2^{\Omega(n)}$ no string z^i selected in the last step of E^f was previously seen by E^f in its simulation of $\tilde{C}^{EX(f,\mathcal{U},\eta)}$; so we assume that this is indeed the case. Since f is a p-biased random function, for each z^i the probability that h_f agrees with f on z^i is at most $1-p$ (recall that $p<\frac{1}{2}$). Thus the probability that E^f outputs 1 is at most the probability that a $(1-p)$-biased coin comes up HEADS at least $(1-\frac{p+\eta-\tau}{2})n$ times in n tosses. Using Chernoff bounds this is at most $1/2^{\Omega(n)}$ (recall that $\eta-\tau<p$ are fixed relative to n so $p-(\eta-\tau)=\Theta(1)$), so the claim is proved. (Claim 21) ■

Now we suppose that f is a p-biased pseudorandom function f_s where s is uniformly chosen in $\{0,1\}^n$. By the definition of p-biased pseudorandomness and Claim 21, for all polynomials Q and all sufficiently large n we have that $\Pr[E^{f_s}$ outputs 1$]<1/Q(n)$ as well. Let $\alpha=\Pr[\tilde{C}^{EX(f_s,\mathcal{U},\eta)}$ outputs a good $h]$, and recall that a good h is an h such that $\Pr[h(x)\neq f_s(x)]\leq\eta-\tau$. Consequently, with probability α, we have that each z^i chosen by E^{f_s} satisfies $h_{f_s}(z^i)=f_s(z^i)$ with probability at least $1-(\eta-\tau)$. Hence with probability α we have that E^{f_s} outputs 1 with probability at least μ, where μ is the probability that a $(1-(\eta-\tau))$-biased coin outputs at least $(1-\frac{p+\eta-\tau}{2})n$ HEADS in n tosses. As before, Chernoff bounds imply that $\mu\geq 1-1/2^{\Omega(n)}$, so consequently $\Pr[E^{f_s}$ outputs 1$]\geq\alpha(1-1/2^{\Omega(n)})$. This proves the claim. (Lemma 20) ■

As a remark, we note that the algorithm A_s is a weak learner for noise rate η and can be modified in a straightforward manner to handle larger noise rates (simply by taking the majority of more examples).

6 Boosting an okay learner to arbitrary accuracy

In this section we present an alternate notion of weak learning, called *okay* learning, and show that the MMM algorithm can be used to efficiently boost any okay learner to arbitrary accuracy in the presence of noise.

To motivate our definition of okay learning, we note that the standard definition of weak learning has some counterintuitive consequences. Consider a scenario in which the target concept $f(x)$ is the Boolean conjunction $x_1\wedge x_2\wedge x_3$ and our hypothesis $h(x)$ is $\neg x_1\wedge\neg x_2\wedge\neg x_3$. Under the uniform distribution we have $\Pr[f(x)\neq h(x)]=1/4$ and hence h is a valid output for a standard weak learner. This is slightly odd since in fact $f(x)$ and $h(x)$ are negatively correlated in a statistical sense, so in some sense a learner which outputs h as a weak hypothesis for f would be a disappointment.

Recall that the balanced distribution $\widehat{\mathcal{D}}$ is obtained by reweighting $\mathcal{D}$ so that $\Pr_{\widehat{\mathcal{D}}}[f(x)=1]=\Pr_{\widehat{\mathcal{D}}}[f(x)=0]=1/2$. We define the *balanced error* of a hypothesis h to be

$$\Pr_{\widehat{\mathcal{D}}}[f(x)\neq h(x)] \quad = \quad \frac{1}{2}\Pr_{\mathcal{D}}[f(x)\neq h(x)\mid f(x)=1]+\frac{1}{2}\Pr_{\mathcal{D}}[f(x)\neq h(x)\mid f(x)=0]. \qquad (3)$$

Similarly, a *noise tolerant γ-okay learner* is an algorithm which, given access to $EX(f,\mathcal{D},\eta)$, outputs a hypothesis h such that $\Pr_{\widehat{\mathcal{D}}}[h(x)\neq f(x)]\leq\frac{1}{2}-\gamma$. The running time is allowed to be polynomial in n, $\frac{1}{1-2\eta}$, $\frac{1}{\delta}$, $\frac{1}{\gamma}$, $\frac{1}{\Pr_{\mathcal{D}}[f(x)=1]}$ and $\frac{1}{\Pr_{\mathcal{D}}[f(x)=0]}$.

While this definition may seem artificially chosen to make our guarantees work, it is actually fairly natural. One observation is that having balanced error $\leq\gamma$ is equivalent to

$$Cov(h,f)\geq 2\gamma Cov(f,f),$$

where, $Cov(f,h) = E_{\mathcal{D}}[f(x)h(x)] - E_{\mathcal{D}}[f(x)]E_{\mathcal{D}}[h(x)]$ is the covariance of f and h. So it is a guarantee that the covariance is positive (equivalently correlation is positive). Another consequence is that $\Pr_{\mathcal{D}}[h(x) = 1|f(x) = 1] > \Pr_{\mathcal{D}}[h(x) = 1]$. In the absence of noise, an okay learning algorithm can be converted to a weak learning algorithm and vice versa. In the presence of noise, an okay learner can be converted to a weak learner.

Given access to a noise tolerant okay learner, we modify the MM algorithm in the following ways:

- As before we calculate p_ℓ according to (2).
- In Step 2 we run the noise-tolerant γ-okay learner using the *unbalanced* conditional distribution $EX(f, \mathcal{D}|_\ell, \eta)$.

As in the MM algorithm we boost until we obtain an h which satisfies $\Pr_{\mathcal{D}}[h(x) \neq f(x)] \leq \epsilon$. We obtain:

Theorem 22 *For any $\epsilon, \delta > 0$, if the above boosting algorithm is run using a noise-tolerant γ-okay learner and a noisy example oracle $EX(f, \mathcal{D}, \eta)$, then it runs for at most poly$(\frac{1}{\gamma}, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{1-2\eta})$ time steps and with probability $1 - \delta$ outputs a hypothesis h satisfying $\Pr_{\mathcal{D}}[h(x) \neq f(x)] \leq \epsilon$.*

Proof: The analysis for boosting a noise-tolerant γ-okay learner is identical to the original noise-free MM analysis. Each hypothesis generated by our noise-tolerant γ-okay learner using an oracle $EX(f, \mathcal{D}, \eta)$ satisfies $\Pr_{\widehat{\mathcal{D}}}[h(x) \neq f(x)] \leq \frac{1}{2} - \gamma$ which is exactly the condition that was used in our noise-free analysis. ∎

We note that an okay learner is equivalent to simply a learner that satisfies Mansour and McAllester notion of "index reduction hypothesis" [15], namely assuming that the algorithm makes progress each step. However, we follow the original spirit of boosting as a method of increasing weak (or okay) to strong. Further work [9], studies in detail these types of okay learners (and even weaker learners), giving such learners for simple and advanced problems.

7 Conclusions

We have given matching upper and lower bounds for boosting in the presence of classification noise. Intuitively, the key to our positive results for the MM algorithm is that changing the label of any example does not change its weight by very much. This property also holds for the earlier decision tree boosting algorithm analyzed by Kearns and Mansour [11], but as mentioned earlier the size of the decision tree could be exponential in $\frac{1}{\gamma}$. While the MM algorithm gives a substantial improvement, the $O(\frac{1}{\gamma^4})$ hypothesis size of the MM algorithm is still larger than the $O(\frac{1}{\gamma^2})$ which other boosting algorithms such as AdaBoost achieve.

Finally, we have defined a noise-tolerant okay learner which can be boosted to arbitrary accuracy in the presence of noise. We hope this will be an aid to designing provably noise-tolerant strong learners, just as the concept of boosting weak learning makes it easier to design provably strong learners.

Follow-up work [9] has extended the analysis of branching program boosting algorithms to different models of noise (probabilistic concepts [12] more similar to statistical regression), giving another theoretical interpretation of noisy boosting.

7.1 Acknowledgments

We would like to thank the anonymous referees and Daphne Koller for helpful comments.

References

[1] J. Aslam and S. Decatur. Specification and simulation of statistical query algorithms for efficiency and noise tolerance. *Journal of Computer and System Sciences*, 56:191–208, 1998.

[2] A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1/2):35–52, 1997.

[3] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

[4] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. The Annals of Statistics, 28:337 – 374, 2000.

[5] Y. Freund. Boosting a weak learning alogrithm by majority. *Information and Computation*, 121(2),256–285, 1995.

[6] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[7] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the Association for Computing Machinery*, 33(4):792–807, 1986.

[8] J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[9] A. Kalai. Learning Monotonic Linear Funcitons. In *Proceedings of the 17th Annual Conference on Learning Theory*, 487-501, 2004.

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

[11] 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.

[12] M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, 382-391, 1990.

[13] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

[14] M. Kearns and U. Vazirani. *An introduction to computational learning theory.* MIT Press, Cambridge, MA, 1994.

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

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

[17] R. Schapire. Theoretical views of boosting. In *Proceedings of the Tenth International Conference on Algorithmic Learning Theory*, pages 12–24, 1999.

A p-biased pseudorandom function families

Let $\ell(\cdot)$ be a polynomial. Recall from [7] that a *pseudorandom function family* is a collection of functions $\{f_s : \{0,1\}^{|s|} \to \{0,1\}^{\ell(|s|)}\}_{s\in\{0,1\}^*}$ with the following two properties:

- (efficient evaluation) There is a deterministic algorithm which, given an n-bit seed s and an n-bit input x, runs in time poly(n) and outputs $f_s(x)$.
- (pseudorandomness) For all polynomials Q, all probabilistic polynomial time oracle algorithms M, and all sufficiently large n, we have

$$\left| \Pr_{F\in\mathcal{F}_n}[M^F(1^n) \text{ outputs } 1] - \Pr_{s\in\{0,1\}^n}[M^{f_s}(1^n) \text{ outputs } 1] \right| < \frac{1}{Q(n)}.$$

  where $\mathcal{F}_n$ is the set of all $2^{\ell(n)2^n}$ functions which map $\{0,1\}^n$ to $\{0,1\}^{\ell(n)}$ (and hence $F \in \mathcal{F}_n$ is a truly random function).

It is well known [7, 8] that pseudorandom function families exist if and only if one-way functions exist.

For $0 < p < 1$, we define a *p-biased pseudorandom function family* to be a family of functions $\{f_s : \{0,1\}^{|s|} \to \{0,1\}\}_{s\in\{0,1\}^*}$ which satisfies the usual "efficient evaluation" property and the following "p-biased pseudorandomness" property:

- (p-biased pseudorandomness) For all polynomials Q, all probabilistic polynomial time oracle algorithms M, and all sufficiently large n, we have

$$\left| \Pr_{F\in\mathcal{F}_{n,p}}[M^F(1^n) \text{ outputs } 1] - \Pr_{s\in\{0,1\}^n}[M^{f_s}(1^n) \text{ outputs } 1] \right| < \frac{1}{Q(n)}.$$

  where $\mathcal{F}_{n,p}$ is the distribution over functions from $\{0,1\}^n$ to $\{0,1\}$ such that each function F has weight $p^{|F^{-1}(1)|}(1-p)^{|F^{-1}(0)|}$. Equivalently, drawing a function $F \in \mathcal{F}_{n,p}$ is done by tossing a p-biased coin for each $x \in \{0,1\}^n$ to determine $F(x)$.

We use the fact that for any $0 < p < 1$, if one-way functions exist then p-biased pseudorandom function families exist. To see this, consider a pseudorandom function family $\{f_s\}$ in which $\ell(n) = n$. Let $\{f'_s\}$ be a family of binary-valued functions defined as follows: $f'_s(x) = 1$ if $f_s(x)$ is one of the first $\lceil p2^n \rceil$ lexicographically ordered strings in $\{0,1\}^n$, and $f'_s(x) = 0$ otherwise. It is straightforward to verify that $\{f'_s\}$ is a p-biased pseudorandom function family.