Martingale Boosting*

Philip M. Long
Google
Mountain View, CA
plong@google.com

Rocco A. Servedio†
Columbia University
New York, NY
rocco@cs.columbia.edu

September 12, 2008

Abstract

Martingale boosting is a simple and easily understood technique with a simple and easily understood analysis. A slight variant of the approach provably achieves optimal accuracy in the presence of misclassification noise.

1 Introduction

Boosting [15, 8] has been an overwhelming practical success. In many applied domains, the best known algorithms use boosting. Nevertheless, some time ago, sensitivity to noise was identified as a weakness of the standard boosting techniques [7, 11, 5].

Heuristics have been proposed to combat this [14, 13]. The heuristics are based on an implicit view that noisy examples tend to be borderline cases: they penalize noisy examples roughly in proportion to how much they deviate from the norm. This view has been seen to be useful, but there are applications in which many examples are not borderline.

Some boosting algorithms have been shown to be provably noise-tolerant [16, 2, 3, 9, 10]. As in classification in general, the main approaches to theory for noise-tolerant boosting can be divided into agnostic/malicious and independent models. In the agnostic/malicious case, essentially nothing is assumed about the noise, except a limit on its rate. This may appear to be more realistic than the alternative in which the labels are assumed to be flipped independently of the sample. However, analysis of agnostic or malicious noise models is by necessity focused on the worst case; typically, in this case, noisy examples are the most extreme elements of the opposite class. Sources involving independent misclassification resemble applied problems more than this. Thus, analysis of learning with independent misclassification noise may be the most effective way to use theory to guide the design of boosting algorithms that are robust to noisy data other than borderline cases.

1.1 Our contribution This paper is about an approach that we call *martingale boosting*. We concentrate on the problem of predicting binary classifications, say 0 and 1. As in many earlier boosting algorithms, learning proceeds incrementally in stages. In each stage, examples are partitioned into bins, and a separate base classifier is chosen for each bin. An example is assigned a bin by counting the number of 1 predictions made by the appropriate base classifiers from earlier rounds.

*This is a full version which contains some proofs omitted from the conference version because of space.

†Supported in part by NSF CAREER award CCF-0347282

The analysis is very simple: it proceeds by thinking of an object to be classified as taking a random walk on the number of base classifiers that predict 1. If the error rates are balanced between false positives and false negatives and are slightly better than random guessing, it is easy to see that, after a few rounds, it is overwhelmingly likely that more than half the steps are in the correct direction: such examples are classified correctly by the boosted classifier.

In some cases, one can promote balanced error rates directly; for example, if decision stumps are used as base classifiers, one can easily adjust the threshold to balance the error rates on the training data. We also show that it is possible to *force* a standard weak learner to produce a classifier with balanced error rates in the cases that we need.

Martingale boosting facilitates noise tolerance by the fact that the probability of reaching a given bin depends on the *predictions* made by the earlier base classifiers, and not on the label of an example. (In particular, it does not depend on the number that are correct or incorrect, as does Boost-by-Majority [6].) The most technical aspect of the paper is to show that the reweighting to force balanced errors can be done while preserving noise-tolerance. Ideas from earlier work by Kalai and Servedio [10] are useful there.

Because it is a simple and easily understood technique that generates highly noise-tolerant algorithms, ideas from martingale boosting appear likely to be practically useful.

1.2 Organization Section 3 gives a high-level description of the architecture of our boosting algorithm and compares it with the three previous boosting algorithms which seem most closely related, namely the Boost-by-Majority algorithm of Freund [6], the branching program booster of Mansour and McAllester [12] and the noise-tolerant booster of Kalai and Servedio [10].

In Section 4 we consider a learning scenario in which the weak learning algorithm is assumed to satisfy a somewhat stronger guarantee than the usual weak learning assumption: the base classifier produced by the weak learner is assumed to achieve accuracy significantly greater than $1/2$ on both positive and negative examples. Working in this scenario lets us highlight the main idea of our new boosting method; we give a very simple proof in Section 4 that our algorithm successfully boosts such a weak learner to arbitrary accuracy.

In Section 5, we show how a slight modification of the algorithm from Section 4 can be used to boost a standard weak learning algorithm, which only meets the usual weak learning criterion, to arbitrarily high accuracy. We discuss computational issues and give an implementation of this algorithm in a framework where the learner has access to an example oracle $EX(c, \mathcal{D})$ in Section 6.

In Section 7, we show how the algorithm of Section 5 can be used to boost a random classification noise tolerant weak learning algorithm to achieve final accuracy $1 - \epsilon$, where ϵ is any value greater than the noise rate η. The modification described in this section is similar to that of Kalai and Servedio [10] who showed how the boosting algorithm of McAllester and Mansour could be modified to achieve tolerance to random classification noise.

2 Preliminaries

Given a target concept $c : X \rightarrow \{0, 1\}$ and a distribution $\mathcal{D}$ over X, we write $\mathcal{D}^+$ to denote the distribution $\mathcal{D}$ restricted to the positive examples $\{x \in X : c(x) = 1\}$. Thus, for any event $S \subseteq \{x \in X : c(x) = 1\}$ we have $\Pr_{\mathcal{D}^+}[x \in S] = \Pr_{\mathcal{D}}[x \in S] / \Pr_{\mathcal{D}}[c(x) = 1]$. Similarly, we write $\mathcal{D}^-$ to denote $\mathcal{D}$ restricted to the negative examples $\{x \in X : c(x) = 0\}$.

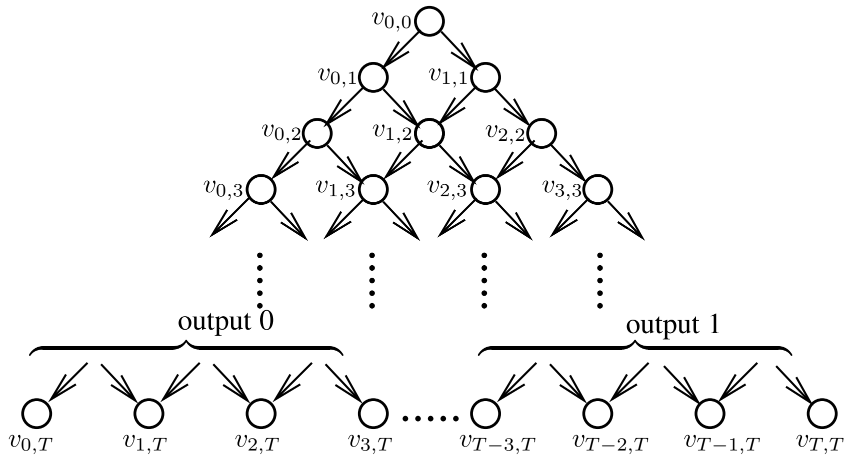

Figure 1: The branching program produced by the boosting algorithm. Each node $v_{i,t}$ is labeled with a 0/1-valued function $h_{i,t}$; left edges correspond to 0 and right edges to 1.

3 High-level structure of the boosting algorithm

The boosting algorithm works in a series of T *stages*. The hypothesis of the boosting algorithm is a layered branching program with $T+1$ layers in a grid graph structure, where layer t has $t+1$ nodes (see Figure 1); we refer to the i-th node from the left in layer t as $v_{i,t}$, where i ranges from 0 to t. For $0 \leq t \leq T-1$, each node $v_{i,t}$ in layer t has two outgoing edges, one left edge (which is labeled with a 0) to node $v_{i,t+1}$ and one right edge (labeled with a 1) to node $v_{i+1,t+1}$. Nodes $v_{i,T}$ in layer T have no outgoing edges.

Before stage t of the boosting algorithm begins, each node at levels $0, \dots, t-1$ has been labeled with a 0/1-valued hypothesis function. We write $h_{i,j}$ to denote the hypothesis function that labels node $v_{i,j}$. In the t-th stage, hypothesis functions are assigned to each of the $t+1$ nodes $v_{0,t}$ through $v_{t,t}$ at level t. Given an example $x \in X$ in stage t, the branching program routes the example by evaluating $h_{0,1}$ on x and then sending the example on the outgoing edge whose label is $h_{0,1}(x)$, i.e. sending it to node $v_{h_{0,1}(x),1}$. The example is routed through successive levels in this way until it reaches level t; more precisely, when example x reaches some node $v_{i,j}$ in level j, it is routed from there via the outgoing edge whose label is $h_{i,j}(x)$ to the node $v_{i+h_{i,j}(x),j+1}$. In this fashion the example x eventually reaches the node $v_{\ell,t}$ after being evaluated on t hypotheses, where ℓ is the number of these t hypotheses which evaluated to 1 on x.

Thus, in the t-th stage of boosting, given an initial distribution $\mathcal{D}$ over examples x, the hypotheses that have been assigned to nodes at levels $0, \dots, t-1$ of the branching program induce $t+1$ different distributions $\mathcal{D}_{0,t}, \dots, \mathcal{D}_{t,t}$ corresponding to the $t+1$ nodes $v_{0,t}, \dots, v_{t,t}$ in layer t (a random draw x from distribution $\mathcal{D}_{i,t}$ is a draw from $\mathcal{D}$ conditioned on x reaching $v_{i,t}$). In the following sections, we will carefully specify just how the hypotheses $h_{0,t}, \dots, h_{t,t}$ are generated to label the nodes $v_{0,t}, \dots, v_{t,t}$ in the t-th stage of boosting; as we will see in Section 5, for the boosting algorithms that work in the standard model, it is *not* the case that $h_{i,t}$ is obtained simply by running the weak learner on distribution $\mathcal{D}_{i,t}$ and using the resulting hypothesis as $h_{i,t}$.

Once all T stages of boosting have been performed, the resulting branching program routes any example x to some node $v_{\ell,T}$ at level T; observe that ℓ is the number of hypotheses that evaluated to 1 out of the T hypotheses that were evaluated on x. The final classifier computed by the branching program is simple: given an example x to classify, if the final node $v_{\ell,T}$ that x reaches has $\ell \geq T/2$ then the output is 1, and otherwise the output is 0.

3.1 Relation to previous boosting algorithms Readers who are familiar with Freund's paper on the Boost-by-Majority algorithm [6] may experience a sense of déjà vu on looking at Figure 1, since a very similar figure appears in [6]. Indeed, both our current boosting scheme and the Boost-by-Majority algorithm can be viewed as routing an example through a branching program which has the graph structure shown in Figure 1, and both boosters work by ultimately predicting 1 or 0 according to whether the majority of T weak hypotheses evaluate to 1 or 0. However, we emphasize that there is a very significant difference between our boosting approach and that of [6]. In Boost-by-Majority, in stage t the weak learning algorithm is only invoked once, using a single distribution $\mathcal{D}_t$ that reweights each examples according to which node $v_{i,t}$ at level t it arrives at. Thus, in Boost-by-Majority there are only T weak hypotheses that are ever generated in the course of boosting, and each node $v_{0,t}, \ldots, v_{t,t}$ is labeled with the same weak hypothesis h_t; the final output is a majority vote over these T hypotheses $h_1, \ldots, h_T$. In contrast, our algorithm invokes the weak learner $t + 1$ separate times in stage t, once for each of the $t + 1$ distinct distributions $\mathcal{D}_{0,t}, \ldots, \mathcal{D}_{t,t}$ corresponding to the nodes $v_{0,t}, v_{1,t}, \ldots, v_{t,t}$. (We remind the reader again that as we will see in Section 5, the hypothesis $h_{i,t}$ is *not* obtained simply by running the weak learner on $\mathcal{D}_{i,t}$ and taking the resulting hypothesis to be $h_{i,t}$.) A total of $T(T + 1)/2$ weak hypotheses are constructed, and any single example x only encounters T of these hypotheses in its path through the branching program.

As we will see, our algorithm has a very simple proof of correctness which seems quite different from the Boost-by-Majority proof. Moreover, the fact that our algorithm constructs a different hypothesis $h_{i,t}$ for each node $v_{i,t}$ seems to play an important role in enabling our boosting algorithm to tolerate random classification noise. We will show in Section 7 that a slight variant of our boosting algorithm can learn to any accuracy rate $1 - \epsilon < 1 - \eta$ in the presence of random classification noise at rate η; no such guarantee is given for Boost-by-Majority or any variant of it that we are aware of in the literature, and we were unable to prove such a guarantee for Boost-by-Majority. It is an interesting question for future work to determine whether Boost-by-Majority actually has (close to) this level of noise tolerance.

Another related algorithm is the "boosting by branching programs" algorithm of Mansour and McAllester [12], which we refer to as the MM algorithm. Kalai and Servedio [10] modified the MM algorithm to obtain a boosting algorithm which is robust in the presence of random classification noise.

Like the Mansour/McAllester boosting algorithm, our booster works by building a branching program. Also, as mentioned earlier, our modification and analysis of this paper's boosting algorithm to achieve random classification noise tolerance will follow the approach of Kalai & Servedio. However, there are significant differences between our boosting algorithm and this earlier work. The algorithm and analysis of [12] and [10] are based on the notion of "purity gain;" a node v is split into two descendents if each of the two labels 0 and 1 is achieved by a nonnegligible fraction of the examples that reach v, and two nodes v and w are merged if the ratio of positive to negative examples within v is similar to the ratio within w. Nodes that are pure (for some $b \in \{0, 1\}$ almost all examples that reach v are labeled with b) are "frozen" (i.e. not split any more) and assigned the label b. In contrast, in our new algorithm the label of a given terminal node in the branching program depends not on the majority vote label of examples that reach that node, but on the majority vote label of the hypotheses that are evaluated on the path to the node. In the analysis of our algorithm, progress is measured not in terms of purity gain achieved by splitting a node, but rather by the amount of "drift" in the right direction that a node imparts to the examples that reach it. (We will see, though, that notions of purity do play a role for efficiency reasons in the example oracle model implementation of the algorithm that we describe in Section 6.)

We note also that the algorithms and analyses of [12, 10] gave little insight into the structure of the branching programs that they create. In contrast, our algorithm yields a well-structured and easily intelligible branching program as shown in Figure 1.

4 Boosting a two-sided weak learner

Let $c : X \to \{0,1\}$ be the target function that we are trying to learn to high accuracy with respect to distribution $\mathcal{D}$ over X. Throughout this section the distributions $\mathcal{D}^+$ and $\mathcal{D}^-$ are defined with respect to c.

Definition 1. *A hypothesis $h : X \to \{0,1\}$ is said to have* two-sided advantage γ with respect to $\mathcal{D}$ *if it satisfies both* $\Pr_{x \in \mathcal{D}^+}[h(x) = 1] \geq \frac{1}{2} + \gamma$ *and* $\Pr_{x \in \mathcal{D}^-}[h(x) = 0] \geq \frac{1}{2} + \gamma$.

Thus such a hypothesis performs noticeably better than random guessing both on positive examples and on negative examples. In this section we will assume that we have access to a *two-sided weak learner* that, when invoked on target concept c and distribution $\mathcal{D}$, outputs a hypothesis with two-sided advantage.

In the next section, we will perform an analysis using the usual assumption of having just a standard weak learner. That analysis can be viewed as reducing that problem to the two-side model studied here. However, results in the two-sided model are arguably interesting in their own right for the following reason. In practice, boosting algorithms are often applied in conjunction with "decision stumps", classifiers that base their predictions on whether individual variables are above or below thresholds. For such classifiers, the threshold can be adjusted so that the training error is balanced between positive and negative examples, with a modest increase in the overall error rate; this promotes finding base classifiers with reasonably small error on both kinds of examples, as is required by the definition in the two-sided model.

We now show how the general boosting framework of Section 3 can be used to boost a two-sided weak learner to high accuracy. This is done very simply: in stage t, at each node $v_{i,t}$ we just run the two-sided weak learner on examples drawn from $\mathcal{D}_{i,t}$ (recall that this is the distribution obtained by filtering $\mathcal{D}$ to accept only those examples that reach node $v_{i,t}$), and use the resulting hypothesis, which has two-sided advantage with respect to $\mathcal{D}_{i,t}$, as the hypothesis function $h_{i,t}$ labeling node $v_{i,t}$. We refer to this boosting scheme as `Basic MartiBoost`.

The idea of the analysis is extremely simple. Let h denote the final branching program that `Basic Martiboost` constructs. We will see that a random example x drawn from $\mathcal{D}^+$ (i.e. a random positive example) is routed through h according to a random walk that is biased toward the right, and a random example x drawn from $\mathcal{D}^-$ is routed through h according to a random walk that is biased toward the left. Since h classifies example x according to whether x reaches a final node $v_{\ell,T}$ with $\ell \geq T/2$ or $\ell < T/2$, this will imply that h has high accuracy on both random positive examples and random negative examples.

So consider a random positive example x (i.e. x is distributed according to $\mathcal{D}^+$). For any node $v_{i,t}$, conditioned on x reaching node $v_{i,t}$ we have that x is distributed according to $(\mathcal{D}_{i,t})^+$. Consequently, by the definition of two-sided advantage we have that x goes from node $v_{i,t}$ to node $v_{i+1,t+1}$ with probability at least $1/2 + \gamma$, so x does indeed follow a random walk biased to the right. Similarly, for any node $v_{i,t}$ a random negative example that reaches node $v_{i,t}$ will proceed to node $v_{i,t+1}$ with probability at least $1/2 + \gamma$, and thus random negative examples follow a random walk biased to the left. Now standard bounds on random walks are easily seen to imply that if $T = O(\frac{\log 1/\epsilon}{\gamma^2})$, then the probability that a random positive example x ends up at a node $v_{\ell,T}$ with $\ell < T/2$ is at most ϵ. The same is true for random negative examples, and thus h has overall accuracy at least $1 - \epsilon$ with respect to $\mathcal{D}$.

In more detail, we have the following theorem:

Theorem 1. *Let $\gamma_0, \gamma_1, \ldots, \gamma_{T-1}$ be any sequence of values between 0 and $1/2$. For each value $t = 0, \ldots, T-1$, suppose that each of the $t+1$ invocations of the weak learner on distributions $\mathcal{D}_{i,t}$ (with $0 \leq i \leq$*

t) yields a hypothesis $h_{i,t}$ which has two-sided advantage γ_t with respect to $\mathcal{D}_{i,t}$. Then the final output hypothesis h that `Basic Martiboost` *computes will satisfy* $\Pr_{x\in\mathcal{D}}[h(x) \neq c(x)] \leq \exp\left(-(\sum_{t=0}^{T-1}\gamma_t)^2/(2T)\right)$.

Proof. As sketched above, we will begin by bounding the error rate on positive examples (a nearly identical proof will work for the negative examples).

For $t = 0, \ldots, T$ we define the integer-valued random variable X_t as follows: given a draw of x from $\mathcal{D}^+$, let i denote the index of the node $v_{i,t}$ that x reaches at level t of the branching program. The value of X_t is i.

Fix $1 \leq t \leq T$ and let us consider the conditional random variable $(X_t|X_{t-1})$. Conditioned on X_{t-1} taking any particular value (i.e. on x reaching any particular node $v_{i,t-1}$), we have that x is distributed according to $(\mathcal{D}_{i,t-1})^+$, and thus we have

$$\mathbf{E}[X_t|X_{t-1}] = X_{t-1} + \Pr_{x\in(\mathcal{D}_{i,t-1})^+}[h_{i,t-1}(x) = 1] \geq X_{t-1} + \frac{1}{2} + \gamma_{t-1}, \tag{1}$$

where the inequality follows from the two-sided advantage of $h_{i,t-1}$.

Now for $t = 0, \ldots, T$ define the random variable Y_t as $Y_t = X_t - \sum_{i=0}^{t-1}(\frac{1}{2} + \gamma_i)$ (so $Y_0 = X_0 = 0$.) Since each possible value of Y_{t-1} corresponds to a unique value of X_{t-1} and vice versa, conditioning on the value of Y_{t-1} is the equivalent conditioning on the value of X_{t-1}. Using inequality (1) we obtain

$$\begin{aligned}
\mathbf{E}[Y_t|Y_{t-1}] &= \mathbf{E}\left[X_t - \sum_{i=0}^{t-1}(\frac{1}{2} + \gamma_i)|Y_{t-1}\right] = \mathbf{E}[X_t|Y_{t-1}] - \sum_{i=0}^{t-1}(\frac{1}{2} + \gamma_i) \\
&\geq X_{t-1} + \frac{1}{2} + \gamma_{t-1} - \sum_{i=0}^{t-1}(\frac{1}{2} + \gamma_i) = X_{t-1} - \sum_{i=0}^{t-2}(\frac{1}{2} + \gamma_i) = Y_{t-1},
\end{aligned}$$

so the sequence of random variables $Y_0, \ldots, Y_T$ is a sub-martingale. We have

$$|Y_t - Y_{t-1}| = |X_t - X_{t-1} - (\frac{1}{2} + \gamma_{t-1})| \leq 1,$$

so by Azuma's inequality for sub-martingales (Theorem 7 in Appendix A) we get $\Pr[Y_T \leq -\lambda] \leq \exp(-\lambda^2/(2T))$. We take $\lambda = \sum_{t=0}^{T-1}\gamma_t$, and observe that $\Pr[Y_T \leq -\lambda] = \Pr[X_T \leq T/2] = \Pr_{x\in\mathcal{D}^+}[h(x) = 0]$ for the final hypothesis h. Thus, we indeed have that the error rate on positive examples is at most $\exp\left(-\frac{(\sum_{t=0}^{T-1}\gamma_t)^2}{2T}\right)$. The same argument shows that $\Pr_{x\in\mathcal{D}^-}[h(x) = 1] \leq \exp\left(-\frac{(\sum_{t=1}^{T}\gamma_t)^2}{2T}\right)$, and we are done. ◻

Note that if we have $\gamma_t \geq \gamma$ for all t, then Theorem 1 gives the familiar bound $\Pr_{x\in\mathcal{D}}[h(x) \neq c(x)] \leq \exp(-\frac{\gamma^2 T}{2})$. We further observe that if all the γ_t values are small and equal to the same value γ, then we can replace the 1 in the upper bound on $|Y_t - Y_{t-1}|$ with a value close to $1/2$; and this gives a final bound in which the constant in the exponent is close to 2 (rather than the current value of $1/2$), which is optimal.

5 Boosting a standard weak learner

We recall the usual definition of a weak learner.

Definition 2. *Given a target function $c : X \to \{0,1\}$ and a distribution $\mathcal{D}$, a hypothesis $h : X \to \{0,1\}$ is said to have* advantage γ with respect to $\mathcal{D}$ *if it satisfies* $\Pr_{x\in\mathcal{D}}[h(x) = c(x)] \geq \frac{1}{2} + \gamma$.

	$c(x)=1$	$c(x)=0$
$h(x)=1$	p	q
$h(x)=0$	$1/2-p$	$1/2-q$

	$c(x)=1$	$c(x)=0$
$h(x)=1, \widehat{h}(x)=1$	$\frac{p}{2r}$	$\frac{q}{2r}$
$h(x)=1, \widehat{h}(x)=0$	$p(1-\frac{1}{2r})$	$q(1-\frac{1}{2r})$
$h(x)=0, \widehat{h}(x)=1$	0	0
$h(x)=0, \widehat{h}(x)=0$	$\frac{1}{2}-p$	$\frac{1}{2}-q$

Table 1: Each table entry gives the probability of the corresponding event under the balanced distribution $\mathcal{D}$.

In this section we will assume that we have access to a standard weak learning algorithm which, when invoked on target concept c and distribution $\mathcal{D}$, outputs a hypothesis h which has advantage γ with respect to $\mathcal{D}$. This is the usual assumption that is made in the study of boosting, and is clearly less demanding than the two-sided weak learner we considered in the previous section. We will show how the `Basic Martiboost` algorithm of the previous section can be modified to boost a standard weak learner to high accuracy.

For clarity of exposition, throughout this section we will consider an abstract version of the boosting algorithm in which all desired probabilities can be obtained exactly (i.e. we do not consider issues of sampling error, etc. here). We will deal carefully with these issues when we describe an example oracle model implementation of the algorithm in Section 6.

5.1 Definitions and an easy lemma Let $c : X \to \{0,1\}$ be a target concept. We say that a distribution $\mathcal{D}$ over X is *balanced* if $\mathcal{D}$ puts equal weight on positive and negative examples, i.e. $\Pr_{x \in \mathcal{D}}[c(x) = 0] = \frac{1}{2}$. Given an arbitrary distribution $\mathcal{D}$ (not necessarily balanced), we write $\widehat{\mathcal{D}}$ to denote the balanced version of $\mathcal{D}$ which is an equal average of $\mathcal{D}^+$ and $\mathcal{D}^-$; i.e. for any $S \subseteq X$ we have $\Pr_{\widehat{\mathcal{D}}}[S] = \frac{1}{2}\Pr_{\mathcal{D}^+}[S] + \frac{1}{2}\Pr_{\mathcal{D}^-}[S]$.

Given a distribution $\mathcal{D}$ over X and a hypothesis $h : X \to \{0,1\}$, we define $\widehat{h}$, the balanced version of h, to be the (probabilistic) version of h described below; the key property of $\widehat{h}$ is that it outputs 0 and 1 equally often under $\mathcal{D}$. Let $b \in \{0,1\}$ be the value that h evaluates to more often, and let $r = \Pr_{x \in \mathcal{D}}[h(x) = b]$ (so $1/2 \le r \le 1$). Given an input $x \in X$, to evaluate $\widehat{h}$ on x we toss a biased coin which comes up heads with probability $\frac{1}{2r}$. If we get heads we output $h(x)$, and if we get tails we output $1-b$. This ensures that $\Pr_{x \in \mathcal{D}}[\widehat{h}(x) = b] = \Pr[\text{coin is heads } \& \ h(x) = b] = \frac{1}{2r} \cdot r = \frac{1}{2}$.

The following simple lemma shows that if we have a weak hypothesis h that has advantage γ relative to a balanced distribution $\mathcal{D}$, then the balanced hypothesis $\widehat{h}$ has advantage at least $\gamma/2$ relative to $\mathcal{D}$.

Lemma 1. *If $\mathcal{D}$ is a balanced distribution and* $\Pr_{\mathcal{D}}[h(x) = c(x)] \ge \frac{1}{2} + \gamma$ *then* $\Pr_{\mathcal{D}}[\widehat{h}(x) = c(x)] \ge \frac{1}{2} + \frac{\gamma}{2}$.

Proof. We may assume without loss of generality that $\Pr_{\mathcal{D}}[h(x) = 1] = r \ge \frac{1}{2}$, i.e. that $b = 1$ in the above discussion. If we let p denote $\Pr_{\mathcal{D}}[h(x) = 1 \ \& \ c(x) = 1]$ and q denote $\Pr_{\mathcal{D}}[h(x) = 1 \ \& \ c(x) = 0]$, so $p + q = r$, then the probabilities for all four possible values of h and c are given in the left side of Table 1. From the definition of $\widehat{h}$ it is straightforward to verify that the probabilities of all eight combinations of values for $h, \widehat{h}$ and c are as given in the right side of Table 1. We thus have that $\Pr_{\mathcal{D}}[\widehat{h}(x) = c(x)] = \frac{p}{2r} + q\left(1 - \frac{1}{2r}\right) + \frac{1}{2} - q = \frac{1}{2} + \frac{p-q}{2r}$. By assumption we have $\Pr_{\mathcal{D}}[h(x) = c(x)] \ge \frac{1}{2} + \gamma$, so from the left side of Table 1 we have $p - q \ge \gamma$. The claim follows since $r \le 1$. $\square$

	$h_{i,t}(x) = 0$	$h_{i,t}(x) = 1$
$c(x) = 0$	p	$1/2 - p$
$c(x) = 1$	$1/2 - p$	p

Table 2: Each table entry gives the probability of the corresponding event under the balanced distribution $\widehat{D_{i,t}}$.

5.2 Boosting a standard weak learner with `MartiBoost` Our algorithm for boosting a standard weak learner, which we call `MartiBoost`, works as follows. In stage t, at each node $v_{i,t}$ we run the weak learning algorithm on the balanced version $\widehat{\mathcal{D}_{i,t}}$ of the distribution $\mathcal{D}_{i,t}$; let $g_{i,t}$ denote the hypothesis that the weak learner returns. The hypothesis $h_{i,t}$ that is used to label $v_{i,t}$ is $h_{i,t} = \widehat{g_{i,t}}$, namely $g_{i,t}$ balanced with respect to the balanced distribution $\widehat{\mathcal{D}_{i,t}}$.

The following lemma plays a key role in our proof of correctness:

Lemma 2. *We have* $\Pr_{(x \in \mathcal{D}_{i,t})^+}[h_{i,t}(x) = 1] \geq \frac{1}{2} + \frac{\gamma}{2}$ *and* $\Pr_{(x \in \mathcal{D}_{i,t})^-}[h_{i,t}(x) = 0] \geq \frac{1}{2} + \frac{\gamma}{2}$.

Proof. Since the original hypothesis $g_{i,t}$ that the weak learner returns when invoked with $\widehat{\mathcal{D}_{i,t}}$ has accuracy at least $\frac{1}{2} + \gamma$ with respect to $\widehat{D_{i,t}}$, by Lemma 1 we have that the balanced hypothesis $h_{i,t}$ has accuracy at least $\frac{1}{2} + \frac{\gamma}{2}$ with respect to $\widehat{D_{i,t}}$. Let p denote $\Pr_{\widehat{D_{i,t}}}[h_{i,t}(x) = c(x) = 0]$. Since $\widehat{D_{i,t}}$ is a balanced distribution and $h_{i,t}$ is a balanced hypothesis, it is easy to see that all four table entries must be as given in Table 2, and thus $\Pr_{\widehat{D_{i,t}}}[h_{i,t}(x) = c(x)] = 2p \geq \frac{1}{2} + \frac{\gamma}{2}$, i.e. $p \geq \frac{1}{4} + \frac{\gamma}{4}$. But since $\widehat{D_{i,t}}$ is an equal mixture of $(\mathcal{D}_{i,t})^+$ and $(\mathcal{D}_{i,t})^-$, this implies that $\Pr_{x \in (\mathcal{D}_{i,t})^+}[h_{i,t}(x) = 1] \geq (\frac{1}{4} + \frac{\gamma}{4})/\frac{1}{2} = \frac{1}{2} + \frac{\gamma}{2}$. We similarly have that $\Pr_{x \in (\mathcal{D}_{i,t})^-}[h_{i,t}(x) = 0] \geq \frac{1}{2} + \frac{\gamma}{2}$, and the lemma is proved. □

With this lemma in hand it is easy to prove correctness of `MartiBoost`:

Theorem 2. *Let $\gamma_0, \gamma_1, \ldots, \gamma_{T-1}$ be any sequence of values between 0 and $1/2$. For each value $t = 0, \ldots, T-1$, suppose that each of the $t+1$ invocations of the weak learner on distributions $\widehat{\mathcal{D}_{i,t}}$ (with $0 \leq i \leq t-1$) yields a hypothesis $g_{i,t}$ which has advantage γ_t with respect to $\widehat{\mathcal{D}_{i,t}}$. Then the final branching program hypothesis h that* `MartiBoost` *constructs will satisfy* $\Pr_{x \in \mathcal{D}}[h(x) \neq c(x)] \leq \exp\left(-\frac{(\sum_{t=1}^{T} \gamma_t)^2}{8T}\right)$.

Proof. The proof is almost identical to the proof of Theorem 1. We define sequences of random variables $X_1, \ldots, X_T$ and $Y_0, \ldots, Y_T$ as before; the only difference is that (i) now we have $E[X_t] \geq \frac{1}{2} + \frac{\gamma_t}{2}$ (by Lemma 2) rather than $E[X_t] \geq \frac{1}{2} + \gamma_t$ as in the earlier proof, and (ii) the randomness is now taken over both the draw of x from $\mathcal{D}^+$ and over the internal randomness of each hypothesis $h_{i,t}$ at each node in the branching program. This loss of a factor of 2 from (i) in the advantage accounts for the different constant (worse by a factor of 4) in the exponent of the bound. □

6 Complexity issues: implementation of `MartiBoost` that works with an example oracle

Thus far we have described and analyzed an abstract version of `MartiBoost` without specifying how the weak learner is actually run on the distribution $\widehat{\mathcal{D}_{i,t}}$ at each node. One approach is to run the boosting algorithm on a fixed sample. In this case all relevant probabilities can be maintained explicitly in a look-up table, and then Theorem 2 bounds the training set accuracy of the `MartiBoost` final hypothesis over this fixed sample.

In this section we describe and analyze an implementation of the algorithm in which the weak learner runs given access to an example oracle $EX(c, \mathcal{D})$. As we will see, this version of the algorithm requires some changes for the sake of efficiency; in particular we will "freeze" the execution of the algorithm at nodes $v_{i,t}$ where it is too expensive to simulate $\widehat{\mathcal{D}_{i,t}}$. We give an analysis of the time and sample complexity of the resulting algorithm which shows that it is computationally efficient and can achieve a high accuracy final hypothesis. Note that the accuracy in this case is measured with respect to the underlying distribution generating the data (and future test data).

6.1 The model We define weak learning in the example oracle $EX(c, \mathcal{D})$ framework as follows:

Definition 3. *Given a target function $c : X \to \{0,1\}$, an algorithm A is said to be a* weak learning algorithm with advantage γ *if it satisfies the following property: for any $\delta > 0$ and any distribution $\mathcal{D}$ over X, if A is given δ and access to $EX(c, \mathcal{D})$ then algorithm A outputs a hypothesis $h : X \to \{0,1\}$ which with probability at least $1 - \delta$ satisfies $\Pr_{x \in \mathcal{D}}[h(x) = c(x)] \geq \frac{1}{2} + \gamma$.*

We let $m_A(\delta)$ denote the running time of algorithm A, where we charge one time step per invocation of the oracle $EX(c, \mathcal{D})$. Thus, if we must run algorithm A using a simulated oracle $EX(c, \mathcal{D}')$ but we only have access to $EX(c, \mathcal{D})$, the runtime will be at most $m_A(\delta)$ times the amount of time it takes to simulate a draw from $EX(c, \mathcal{D}')$ given $EX(c, \mathcal{D})$.

6.2 An idealized version of the oracle algorithm We now describe the version of `MartiBoost` designed to work with a sampling oracle in more detail; we call this algorithm `Sampling Martiboost`, or `SMartiBoost`. While this algorithm is intended to work with random examples, to keep the focus clear on the main ideas, let us continue for a while to assume that all required probabilities can be computed exactly. In Section 6.3 we will show that the analysis still holds if probabilities are estimated using a polynomial-size sample.

For convenience, we will use r to denote all of the random bits used by all the hypotheses $h_{i,t}$. It is convenient to think of r as an infinite sequence of random bits that is determined before the algorithm starts and then read off one at a time as needed by the algorithm (though the algorithm will use only polynomially many of them).

In stage t of `SMartiBoost`, all nodes at levels $t' < t$ have been labeled and the algorithm is labeling nodes $v_{0,t}, \ldots, v_{t,t}$. Let $p_{i,t}$ denote $\Pr_{x \in \mathcal{D}, r}[x \text{ reaches } v_{i,t}]$. For each $b \in \{0,1\}$, let $p^b_{i,t}$ denote $\Pr_{x \in \mathcal{D}, r}[x$ reaches $v_{i,t}$ and the label of x is $b]$, so $p_{i,t} = p^0_{i,t} + p^1_{i,t}$. In stage t, `SMartiBoost` does the following for each node $v_{i,t}$:

1. If $\min_{b \in \{0,1\}} p^b_{i,t} < \frac{\epsilon}{T(T+1)}$, then the algorithm "freezes" node $v_{i,t}$ by labeling it with the bit $(1-b)$ and making it a terminal node with no outgoing edges (so any example x which reaches $v_{i,t}$ will be assigned label $(1-b)$ by the branching program hypothesis).
2. Otherwise, we have $\min_{b \in \{0,1\}} p^b_{i,t} \geq \frac{\epsilon}{T(T+1)}$. In this case `SMartiBoost` works just like `MartiBoost`: it runs the weak learning algorithm on the balanced version $\widehat{\mathcal{D}_{i,t}}$ of $\mathcal{D}_{i,t}$ to obtain a hypothesis $g_{i,t}$, and it labels $v_{i,t}$ with $h_{i,t} = \widehat{g_{i,t}}$, which is $g_{i,t}$ balanced with respect to $\widehat{\mathcal{D}_{i,t}}$.

The idea is that each node which is "frozen" in step (1) above contributes at most $\frac{\epsilon}{T(T+1)}$ to the error of the final branching program hypothesis; since there are at most $T(T+1)/2$ many nodes in the branching program, the total error induced by all frozen nodes is at most $\frac{\epsilon}{2}$. On the other hand, for any node $v_{i,t}$ which satisfies condition (2) and is not frozen, the expected number of draws from $EX(c, \mathcal{D})$ that are required to simulate a draw from $EX(c,\widehat{\mathcal{D}_{i,t}})$ is $O(\frac{T^2}{\epsilon})$, and thus we can indeed run the weak learner efficiently on the desired distributions. (We discuss computational efficiency in more detail in the next subsection where we take sampling issues into account.)

The following theorem establishes correctness of `SMartiBoost`:

Theorem 3. *Let $T = \frac{8\ln(2/\epsilon)}{\gamma^2}$. Suppose that each time it is invoked on some distribution $\widehat{\mathcal{D}_{i,t}}$, the weak learner outputs a hypothesis that has advantage γ with respect to $\widehat{\mathcal{D}_{i,t}}$. Then the final branching program hypothesis h that* `SMartiBoost` *constructs will satisfy* $\Pr_{x\in\mathcal{D}}[h(x)\neq c(x)]\leq\epsilon$.

Proof. Given an unlabeled instance $x\in X$ and a particular setting r of the random bits for each of the (randomized) hypotheses $h_{i,t}$ labeling nodes of the branching program, we say that (x,r) *freezes at node* $v_{i,t}$ if the path through the branching program that x takes under randomness r causes it to terminate at a node $v_{i,t}$ with $t<T$ (i.e. at a node $v_{i,t}$ which was frozen by `SMartiBoost`). We have

$$\begin{aligned}\Pr[h(x)\neq c(x)] &= \Pr[h(x)\neq c(x)\ \&\ (x,r)\text{ freezes}]+\Pr[h(x)\neq c(x)\ \&\ (x,r)\text{ does not freeze}]\\ &\leq \frac{\epsilon}{2}+\Pr[h(x)\neq c(x)\ \&\ (x,r)\text{ does not freeze}].\end{aligned}$$

where the probabilities, as in the proof of Theorem 2, are taken over the draw of x from $\mathcal{D}$ and the choice of r.

It remains to show that $\Pr[h(x)\neq c(x)\ \&\ (x,r)\text{ does not freeze}]\leq\frac{\epsilon}{2}$. As before, we first will show that $\Pr_{x\in\mathcal{D}^+}[h(x)\neq c(x)\ \&\ (x,r)\text{ does not freeze}]$ is at most $\frac{\epsilon}{2}$; the negative examples can be handled similarly.

To show that $\Pr_{x\in\mathcal{D}^+}[h(x)\neq c(x)\ \&\ (x,r)\text{ does not freeze}]\leq\frac{\epsilon}{2}$, we consider a slightly different random process than in the proof of Theorem 2. For $t=0,\dots,T$ we now define integer-valued random variables X'_t as follows. Given a draw of x from $\mathcal{D}^+$ and a random choice of r,

- If (x,r) does not freeze at any node $v_{j,t'}$ with $t'<t$, then X'_t takes value i, where i denotes the index of the node $v_{i,t}$ that x reaches under randomess r at level t of the branching program;
- If (x,r) freezes at some node $v_{j,t'}$ with $t'<t$, then let k be the sum of $t-t'$ many independent 0/1 coin tosses each of which comes up 1 with probability $\frac{1}{2}+\frac{\gamma}{2}$. The value of X'_t is $j+k$.

(This part of the proof is reminiscent of [3].) It is clear that

$$E[X'_t\ |X'_{t-1}\ \&\ (x,r)\text{ freezes at some node }v_{j,t'}\text{ with }t'<t]=X'_{t-1}+\frac{1}{2}+\frac{\gamma}{2}.$$

On the other hand, if (x,r) does not freeze at any such node, then conditioned on x reaching any particular node $v_{i,t-1}$ under randomness r we have that x is distributed according to $(\mathcal{D}_{i,t-1})^+$. It follows from Lemma 2 that

$$E[X'_t\ |X'_{t-1}\ \&\ (x,r)\text{ freezes at no node }v_{j,t'}\text{ with }t'\leq t]\geq X'_{t-1}+\frac{1}{2}+\frac{\gamma}{2},$$

and thus overall we have $E[X'_t|X'_{t-1}]\geq\frac{1}{2}+\frac{\gamma}{2}$.

Now similar to the proof of Theorem 1, for $t=0,\dots,T$ let the random variable Y'_t be defined as $Y'_t=X'_t-t(\frac{1}{2}+\gamma)$. As in the earlier proof we have that $Y_0,\dots,Y_T$ is a sub-martingale with bounded differences, so Azuma's inequality for sub-martingales gives $\Pr\left[Y'_T\leq-\gamma T/2\right]\leq\exp\left(-\frac{\gamma^2T}{8}\right)$. Now recall that if (x,r) never freezes, then the prediction $h(x)$ is determined by the majority of the values of $h_{i,t}(x)$ obtained from hypotheses $h_{i,t}$ encountered in its path through the branching program. Thus, in the particular case of positive examples, $\Pr_{x\in\mathcal{D}^+,r}[h(x)\neq c(x)\ \&\ (x,r)\text{ does not freeze}]\leq\Pr\left[X'_T\leq T/2\right]=\Pr[Y'_t\leq-\gamma T/2]$. Applying the inequality from above, bounding negative examples similarly, and recalling our choice of T, we have that $\Pr[h(x)\neq c(x)\ \&\ (x,r)\text{ does not freeze}]\leq\frac{\epsilon}{2}$ and the theorem is proved. ☐

6.3 Dealing with sampling error In this section we remove the assumptions that we know all required probabilities exactly, by showing that sufficiently accurate estimates of them can be obtained efficiently. We do not belabor details since sampling error analyses such as the one we now present are quite standard.

Consider some node $v_{i,t}$. While we cannot determine precisely whether $p^b_{i,t}$ is greater than $\frac{\epsilon}{T(T+1)}$ or not, standard multiplicative Chernoff bounds show that we can compare it up to a constant multiple which is good enough for our purposes:

Fact 3. *With $O(\frac{T^2 \log \frac{1}{\delta}}{\epsilon})$ many draws to $EX(c, \mathcal{D})$, we can obtain an estimate $\tilde{p}^b_{i,t}$ of $p^b_{i,t}$ which satisfies the following:*

- *If $p^b_{i,t} < \frac{\epsilon}{2T(T+1)}$, then with probability $1-\delta$, we have that $\tilde{p}^b_{i,t}$ is at most $\frac{3\epsilon}{4T(T+1)}$; and*
- *If $p^b_{i,t} > \frac{\epsilon}{T(T+1)}$, then with probability $1-\delta$, we have that $\tilde{p}^b_{i,t}$ is at least $\frac{3\epsilon}{4T(T+1)}$.*

Thus `SMartiBoost` will actually decide whether or not to freeze a node $v_{i,t}$ by checking whether $\tilde{p}^b_{i,t}$ exceeds $\frac{3\epsilon}{4T(T+1)}$. If $\min_{b\in\{0,1\}} p^b_{i,t}$ is not too small then we can efficiently simulate draws from $EX(c, \widehat{\mathcal{D}_{i,t}})$ (by tossing a coin to decide whether $EX(c, \widehat{D_{i,t}})$ will give a positive or negative example, and then drawing from $EX(c, \mathcal{D})$ until it gives an example with the correct label which reaches $v_{i,t}$):

Fact 4. *If $\min_{b\in\{0,1\}} p^b_{i,t} \geq \frac{\epsilon}{2T(T+1)}$, then given access to $EX(c, \mathcal{D})$ we can with probability $1-\delta$ perfectly simulate a draw from $EX(c, \widehat{\mathcal{D}_{i,t}})$ by making at most $O(\frac{T^2 \log \frac{1}{\delta}}{\epsilon})$ many draws from $EX(c, \mathcal{D})$.*

Once we have run the weak learning algorithm with $EX(c, \widehat{\mathcal{D}_{i,t}})$ and it has given us its hypothesis $g_{i,t}$, we need to construct $h_{i,t}$, the balanced version of this hypothesis with respect to $\widehat{\mathcal{D}_{i,t}}$. In order to do this perfectly as in Section 5.1, we would need the exact value of $r = \Pr_{x\in\widehat{\mathcal{D}_{i,t}}}[g_{i,t}(x) = b] \geq \frac{1}{2}$. While this exact value is not available to us, the following straightforward generalization of Lemma 1 shows that an approximate value is good enough for our needs:

Lemma 5. *Let $r' \in [0,1]$ be any value such that $|r - r'| \leq \frac{\gamma}{4}$ where r is as described above. Let $h'_{i,t}$ denote the "balanced" hypothesis obtained from $g_{i,t}$ as in Section 5.1 but using r' instead of r. If $\Pr_{\widehat{\mathcal{D}_{i,t}}}[g_{i,t}(x) = c(x)] \geq \frac{1}{2} + \gamma$, then $\Pr_{\widehat{\mathcal{D}_{i,t}}}[h'_{i,t}(x) = c(x)] \geq \frac{1}{2} + \frac{\gamma}{4}$.*

Standard bounds on sampling show that we can obtain an estimate r' of r which is accurate to within an additive $\pm\frac{\gamma}{4}$ with probability $1-\delta$ by making at most $O(\frac{\log(1/\delta)}{\gamma^2})$ many draws from $EX(c, \widehat{\mathcal{D}_{i,t}})$. Putting all the pieces together, we have the following theorem which shows that `SMartiBoost` is correct and efficient (we use $\tilde{O}$ notation to hide polylogarithmic factors, and ignore the dependences on δ – which are everywhere polylogarithmic – throughout for the sake of readability):

Theorem 4. *Let $T = \Theta(\frac{\log(1/\epsilon)}{\gamma^2})$. If A is a weak learning algorithm that requires s_A many examples to construct a γ-advantage hypothesis, then* `SMartiBoost` *makes $O(s_A) \cdot \tilde{O}(\frac{1}{\epsilon}) \cdot poly(\frac{1}{\gamma})$ many calls to $EX(c, \mathcal{D})$ and with probability $1-\delta$ outputs a final hypothesis h that satisfies $\Pr_{x\in\mathcal{D}}[h(x) \neq c(x)] \leq \epsilon$.*

Proof. A straightforward union bound over all failure probabilities (for each call of the weak learning algorithm and for each sampling-based estimate), together with Theorem 3, establishes correctness. For the sample complexity bound, consider a single step of the algorithm when it is processing some node $v_{i,t}$. Estimating $p^b_{i,t}$ takes $\tilde{O}(\frac{T^2}{\epsilon})$ many draws by Fact 3, and then running the weak learner with $EX(c, \widehat{\mathcal{D}_{i,t}})$ takes

$\tilde{O}(\frac{s_A T^2}{\epsilon})$ many draws from $EX(c, \mathcal{D})$ by Fact 4. Estimating r at $v_{i,t}$ to "balance" the hypothesis $g_{i,t}$ that the weak learner returns takes $\tilde{O}(\frac{1}{\gamma^2})$ many draws from $EX(c, \widehat{\mathcal{D}_{i,t}})$, i.e. $\tilde{O}(\frac{T^2}{\gamma^2 \epsilon})$ many draws from $EX(c, \mathcal{D})$. Thus, dealing with a single node $v_{i,t}$ takes $\tilde{O}(\frac{s_A T^2}{\epsilon} + \frac{T^2}{\gamma^2 \epsilon})$ many draws from $EX(c, \mathcal{D})$. Since there are $O(T^2)$ many nodes in the branching program and $T = O(\frac{\log 1/\epsilon}{\gamma^2})$ we obtain the claimed bound. □

7 A noise-tolerant version of `SMartiBoost`

In this section we show how the `SMartiBoost` algorithm can be modified to withstand random classification noise. We follow the approach of Kalai & Servedio [10], who showed how the MM branching program boosting algorithm of Mansour and McAllester can be modified to withstand random classification noise.

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

Recall the definition of noise-tolerant weak learning:

Definition 4. *Given a target function $c : X \to \{0, 1\}$, an algorithm A is said to be a* noise-tolerant weak learning algorithm with advantage γ *if it satisfies the following property: for any $\delta > 0$ and any distribution $\mathcal{D}$ over X, if A is given δ and access to a noisy example oracle $EX(c, \mathcal{D}, \eta)$ where $0 \le \eta < \frac{1}{2}$, then A runs in time poly$(\frac{1}{1-2\eta}, \frac{1}{\delta})$ and with probability at least $1 - \delta$ A outputs a poly$(\frac{1}{\delta}, \frac{1}{\gamma}, \frac{1}{1-2\eta})$-time evaluable hypothesis h such that* $\Pr_{x \in \mathcal{D}}[h(x) = c(x)] \ge \frac{1}{2} + \gamma$.

Ideally, we would like a boosting algorithm that can convert any noise-tolerant weak learning algorithm into a noise-tolerant strong learning algorithm that can achieve any arbitrarily low error rate $\epsilon > 0$. However, Kalai and Servedio showed that in general it is not possible to boost the error rate ϵ down below the noise rate η.[1] They showed that a modification of the MM boosting algorithm, which they called the MMM (Modified Mansour-McAllester) algorithm, can achieve any error rate $\epsilon = \eta + \tau$ in time polynomial in $\frac{1}{\tau}$ and the other relevant parameters. We now show that a modification of `SMartiBoost` has the same property.

Throughout this section we assume that the value of η, the random classification noise rate, is known in advance to the boosting algorithm. As described in [10] this assumption can be removed via a standard approach of guessing different values for the noise rate and using the best outcome.

For ease of presentation, in Section 7.1 we give the noise-tolerant martingale boosting algorithm under the assumption that all required probabilities are obtained exactly. In Section 7.2 we deal with issues of estimating these probabilities via sampling and bound the sample complexity of the algorithm.

7.1 Boosting a noise-tolerant weak learner Roughly speaking, the reason why `SMartiBoost` can be easily modified to withstand random classification noise is because in each stage t of boosting the label b of a labeled example (x, b) plays only a limited role in the reweighting that the example experiences. Since this role is limited, it is possible for us to efficiently simulate the distributions that the weak learner requires at each stage of boosting and thus for the overall boosting process to succeed.

More precisely, as a labeled example (x, b) proceeds through levels $0, \ldots, t-1$ of the branching program in stage t, the path it takes is completely independent of b. Thus, given a source $EX(c, \mathcal{D}, \eta)$ of noisy

[1] They showed that if cryptographic one-way functions exist, then there is no efficient "black-box" boosting algorithm that can always achieve a final error rate $\epsilon < \eta$. A black-box boosting algorithm is a boosting algorithm that can run the weak learning algorithm in a black-box fashion but cannot "inspect the code" of the weak learner. All known boosting algorithms are black-box boosters. See [10] for more discussion.

examples, the distribution of examples that arrive at a particular node $v_{i,t}$ is precisely $EX(c, \mathcal{D}_{i,t}, \eta)$. Once a labeled example (x, b) arrives at some node $v_{i,t}$, though, it is clear that the label b must be consulted in the "rebalancing" of the distribution $\mathcal{D}_{i,t}$ to obtain distribution $\widehat{\mathcal{D}_{i,t}}$. More precisely, the labeled examples that reach node $v_{i,t}$ are distributed according to $EX(c, \mathcal{D}_{i,t}, \eta)$, but in order to use `SMartiBoost` with a noise-tolerant weak learner we must simulate the *balanced* distribution $\widehat{\mathcal{D}_{i,t}}$ corrupted with random classification noise, i.e. $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$. (As we show below, it turns out that η' need not necessarily be the same as η; it is okay to have a higher noise rate η' for the balanced oracle as long as η' is not too close to $\frac{1}{2}$.) The following lemma from [10] shows that it is possible to do this:

Lemma 6. *[Lemma 7 of [10]] Let $\tau > 0$ be any value satisfying $\eta + \frac{\tau}{2} < \frac{1}{2}$. Suppose we have access to $EX(c, \mathcal{D}, \eta)$. Let ρ denote $\Pr_{x \in \mathcal{D}}[c(x) = 1]$. Suppose that $\eta + \frac{\tau}{2} \leq \rho \leq \frac{1}{2}$ (the case where $\eta + \frac{\tau}{2} \leq 1 - \rho \leq \frac{1}{2}$ is completely analogous). Consider the following rejection sampling procedure: given a draw (x, b) from $EX(c, \mathcal{D}, \eta)$,*

- *if $b = 0$ then with probability $p_r = \frac{1-2\rho}{1-\rho-\eta}$ reject (x, b), and with probability $1 - p_r = \frac{\rho-\eta}{1-\rho-\eta}$ set $b' = b$ and accept (x, b');*
- *if $b = 1$ then set b' to $1 - b$ with probability $p_f = \frac{(1-2\rho)\eta(1-\eta)}{(1-\rho-\eta)(\rho+\eta-2\rho\eta)}$ (and set b' to b with probability $1 - p_f$), and accept (x, b').*

Given a draw from $EX(c, \mathcal{D}, \eta)$, with probability $p_{rej} := \frac{(1-2\rho)(\rho\eta+(1-\rho)(1-\eta))}{1-\rho-\eta}$ this procedure rejects, and with probability $1 - p_{rej} = \frac{2(1-2\eta)(1-\rho)\rho}{1-\rho-\eta}$ the procedure accepts. Moreover, if the procedure accepts, then the (x, b') that it accepts is distributed according to $EX(c, \widehat{\mathcal{D}}, \eta')$ where $\eta' = \frac{1}{2} - \frac{\rho-\eta}{2(\rho+\eta-2\rho\eta)}$.

So the noise-tolerant version of `SMartiBoost`, which we call `Noise-Tolerant SMartiBoost`, works in the following way. As in Section 6.2 let $p_{i,t}$ denote $\Pr_{x \in \mathcal{D}, r}[x \text{ reaches } v_{i,t}]$. For $b = 0, 1$ let $q_{i,t}^b$ denote $q_{i,t}^b = \Pr_{x \in \mathcal{D}, r}[c(x) = b \mid x \text{ reaches } v_{i,t}] = \Pr_{x \in \mathcal{D}_{i,t}, r}[c(x) = b]$, so $q_{i,t}^0 + q_{i,t}^1 = 1$. The boosting algorithm (which takes as input a parameter $\tau > 0$, where $\eta + \tau$ is the desired final accuracy of the hypothesis; we assume WLOG that $\eta + \tau < \frac{1}{2}$) proceeds in stage t as follows: at each node $v_{i,t}$,

1. If $p_{i,t} < \frac{2\tau}{3T(T+1)}$, then the algorithm "freezes" node $v_{i,t}$ by labeling it with an arbitrary bit and making it a terminal node with no outgoing edges.
2. Otherwise, if $\min_{b \in \{0,1\}} q_{i,t}^b < \eta + \frac{\tau}{3}$, then the algorithm "freezes" node $v_{i,t}$ by labeling it with the bit $(1 - b)$ and making it a terminal node with no outgoing edges.
3. Otherwise the algorithm runs the noise-tolerant weak learner using $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$ as described in Lemma 6 to obtain a hypothesis $g_{i,t}$. The balanced version of this hypothesis (balanced with respect to $\widehat{\mathcal{D}_{i,t}}$), which we call $h_{i,t}$, is used to label node $v_{i,t}$.

Theorem 5. *Let $T = \frac{8\ln(3/\tau)}{\gamma^2}$. Suppose that each time it is invoked with some oracle $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$, the weak learner outputs a hypothesis $g_{i,t}$ that satisfies $\Pr_{x \in \widehat{\mathcal{D}_{i,t}}}[g_{i,t}(x) = c(x)] \geq \frac{1}{2} + \gamma$. Then the final branching program hypothesis h that* `Noise-Tolerant SMartiBoost` *constructs will satisfy $\Pr_{x \in \mathcal{D}}[h(x) \neq c(x)] \leq \eta + \tau$.*

Proof. As in the proof of Theorem 3, given an unlabeled instance $x \in X$ and a particular setting r of the random bits for each of the (randomized) hypotheses $h_{i,t}$ labeling nodes of the branching program, we say that (x, r) *freezes at node* $v_{i,t}$ if the path through the branching program that x takes under randomness r causes it to terminate at a node $v_{i,t}$ with $t < T$ (i.e. at a node $v_{i,t}$ which was frozen by `Noise-Tolerant SMartiBoost`). We say that a node $v_{i,t}$ is *negligible* if $p_{i,t} < \frac{2\tau}{3T(T+1)}$. We have

$$\begin{aligned}\Pr[h(x) \neq c(x)] \quad = \quad & \Pr[h(x) \neq c(x) \;\&\; (x,r) \text{ does not freeze}] + \\ & \Pr[h(x) \neq c(x) \;\&\; (x,r) \text{ freezes at a negligible node}] + \\ & \Pr[h(x) \neq c(x) \;\&\; (x,r) \text{ freezes at a non-negligible node}].\end{aligned}$$

Since (x, r) reaches a given negligible node $v_{i,t}$ with probability at most $\frac{2\tau}{3T(T+1)}$ and there are at most $T(T+1)/2$ many negligible nodes, $\Pr[h(x) \neq c(x) \&\ (x,r)$ freezes at a negligible node$]$ is at most $\frac{\tau}{3}$. Consequently $\Pr[h(x) \neq c(x)]$ is at most $\frac{\tau}{3} + \Pr[h(x) \neq c(x) \;\&\; (x,r)$ does not freeze$]$ plus

$$\sum_{i,t\,:\,v_{i,t} \text{ is non-negligible}} \Pr[h(x) \neq c(x) |\ (x,r) \text{ freezes at } v_{i,t}] \cdot \Pr[(x,r) \text{ freezes at } v_{i,t}]. \qquad (2)$$

Since $\Pr[h(x) \neq c(x) \mid (x,r)$ freezes at $v_{i,t}]$ equals $\Pr_{x \in \mathcal{D}_{i,t},r}[h(x) \neq c(x)]$, by the fact that the algorithm freezes $v_{i,t}$ if $\min_{b \in \{0,1\}} q^b_{i,t} < \eta + \frac{\tau}{3}$ (case (2) above), we have that (2) is at most $\eta + \frac{\tau}{3}$. Thus

$$\Pr[h(x) \neq c(x)] \leq \Pr[h(x) \neq c(x) \;\&\; (x,r) \text{ does not freeze}] + \eta + \frac{2\tau}{3}.$$

so it remains to show that $\Pr[h(x) \neq c(x) \;\&\; (x,r)$ does not freeze$]$ is at most $\frac{\tau}{3}$. The proof of this is identical to the proof that $\Pr[h(x) \neq c(x) \;\&\; (x,r)$ does not freeze$] \leq \frac{\epsilon}{2}$ in the proof of Theorem 3 but now with $\frac{\tau}{3}$ in place of $\frac{\epsilon}{2}$. ☐

7.2 Complexity issues for implementation of `Noise-Tolerant SMartiBoost` In this section we remove the assumptions that we know all required probabilities exactly, by showing that sufficiently accurate estimates of them can be obtained efficiently via a polynomial amount of sampling. (For simplicity, throughout the discussion we continue to assume that we know the exact value of η as mentioned at the start of Section 7.)

A direct analogue of Fact 3 shows that $O(\frac{T^2 \log 1/\delta}{\tau})$ many draws to $EX(c, \mathcal{D}, \eta)$ suffice to give an estimate $\tilde{p}_{i,t}$ of $p_{i,t} = \Pr_{x \in \mathcal{D},r}[x$ reaches $v_{i,t}]$ such that

- If $p_{i,t} < \frac{\tau}{3T(T+1)}$, then with probability $1 - \delta$, we have that $\tilde{p}_{i,t}$ is at most $\frac{\tau}{2T(T+1)}$; and
- If $p_{i,t} > \frac{2\tau}{3T(T+1)}$, then with probability $1 - \delta$, we have that $\tilde{p}_{i,t}$ is at least $\frac{\tau}{2T(T+1)}$.

Thus `Noise-Tolerant SMartiBoost` will decide whether or not to freeze a node $v_{i,t}$ on the grounds of its being negligible (case (1)) by checking whether $\tilde{p}_{i,t}$ exceeds $\frac{\tau}{2T(T+1)}$.

For case (2), if $p_{i,t}$ is not too small then we can efficiently simulate draws from $EX(c, \mathcal{D}_{i,t}, \eta)$, and thus can estimate $q^b_{i,t}$ efficiently:

Fact 7. *If $p_{i,t} \geq \frac{\tau}{3T(T+1)}$, then $O(\frac{T^2 \log 1/\delta}{\tau^3(1-2\eta)^2})$ draws from $EX(c, \mathcal{D}, \eta)$ suffice to give an estimate $\tilde{q}^b_{i,t}$ of $q^b_{i,t}$ such that*

- *If $q^b_{i,t} < \eta + \frac{\tau}{6}$, then with probability $1 - \delta$, we have that $\tilde{q}_{i,t}$ is at most $\eta + \frac{\tau}{4}$; and*

- *If $p^b_{i,t} > \eta + \frac{\tau}{3}$, then with probability $1-\delta$, we have that $\tilde{q}_{i,t}$ is at most $\eta + \frac{\tau}{4}$.*

Proof. It is easy to see that

$$q^b_{i,t} = \Pr_{x \in \mathcal{D}_{i,t}}[c(x) = b] = \frac{\Pr_{(x,y)\in EX(c,\mathcal{D}_{i,t},\eta)}[y = b] - \eta}{1-2\eta}.$$

Thus in order to estimate $q^b_{i,t}$ to within an additive $\Theta(\tau)$, we must estimate $\Pr_{(x,y)\in EX(c,\mathcal{D}_{i,t},\eta)}[y = b]$ to within an additive $\Theta(\tau(1-2\eta))$. This requires $\Theta(\frac{1}{\tau^2(1-2\eta)^2})$ many draws from $EX(c, \mathcal{D}_{i,t}, \eta)$, and since $p_{i,t} = \Omega(\frac{\tau}{T^2})$, there is an overhead of $\Theta(\frac{T^2}{\tau})$ to simulate each draw from $EX(c, \mathcal{D}_{i,t}, \eta)$ using $EX(c, \mathcal{D}, \eta)$. □

Now we consider case (3), which requires us to use the rejection sampling procedure of Lemma 6. We first note that if the exact value of $\rho = \Pr_{x\in\mathcal{D}_{i,t}}[c(x) = 1]$ were available, then the probability p_{rej} of rejecting would not be too high (so it would indeed be possible to efficiently simulate $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$):

Claim 8. *The rejection probability $p_{rej} := \frac{(1-2\rho)(\rho\eta+(1-\rho)(1-\eta))}{1-\rho-\eta}$ from Lemma 6 satisfies $p_{rej} \leq 1-(\eta+\frac{\tau}{2})$.*

Proof. Recall that $\eta + \frac{\tau}{2} \leq \rho \leq \frac{1}{2}$. ¿From this it is straightforward to verify that we have

$$\eta + \frac{\tau}{2} \leq 2(\eta + \frac{\tau}{2})(1 - \eta - \frac{\tau}{2}) \leq 2\rho(1-\rho) \leq \frac{2\rho(1-\rho)(1-2\eta)}{1-\rho-\eta}.$$

The claim follows since $p_{rej} = \frac{(1-2\rho)(\rho\eta+(1-\rho)(1-\eta))}{1-\rho-\eta}$ equals $1 - \frac{2\rho(1-\rho)(1-2\eta)}{1-\rho-\eta}$. □

This claim implies that given $EX(c, \mathcal{D}_{i,t}, \eta)$, if ρ were known exactly then we could sample from $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$ with a slowdown of at most $\frac{1}{\eta+\tau/2} < \frac{2}{\tau}$. Moreover, if the exact value of ρ were available, then the noise rate η' would not be too close to $\frac{1}{2}$ (so it would be possible to efficiently run the weak learner using $EX(c, \mathcal{D}_{i,t}, \eta)$). This is because (as shown in Lemma 7 of [10]) the noise rate η' from Lemma 6 satisfies $\eta' = \frac{1}{2} - \frac{\rho-\eta}{2(\rho+\eta-2\rho\eta)} \leq \frac{1}{2} - \frac{\tau}{4}$. Since the weak learner runs in time poly$(\frac{1}{1-2\eta'}, \frac{1}{\delta})$ when invoked on $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$, the runtime for each invocation of the weak learner would be bounded by poly$(\frac{1}{\tau}, \frac{1}{\delta})$.

So if the exact value of $\rho = \Pr_{x\in\mathcal{D}_{i,t}}[c(x) = 1]$ were available, then using $EX(c, \mathcal{D}, \eta)$ we could simulate $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$ and run the weak learner at a given node $v_{i,t}$ in poly$(\frac{T}{\tau}, \frac{1}{\delta})$ time overall (there is an $O(\frac{T}{\tau^2})$ factor because of the overhead to simulate $EX(c, \mathcal{D}_{i,t}, \eta)$ from $EX(c, \mathcal{D}, \eta)$). But of course, we do not have the exact value of the probability ρ that Lemma 6 requires; instead we must use an estimate ρ' of ρ that we obtain by sampling. Since ρ' is not exactly p, the oracle we simulate may not be exactly balanced, and it may have slightly different noise rates for positive and negative examples; but as we now show, the deviation from $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$ can be made so small that the weak learner is unaffected.

It is easy to see that $\rho = \frac{\tilde{\rho}-\eta}{1-2\eta}$ where $\tilde{\rho}$ equals $\Pr_{(x,b)\in EX(c,\mathcal{D}_{i,t},\eta)}[b = 1]$. Thus we can estimate ρ to within an additive $\pm\xi$ by estimating $\tilde{\rho}$ from $EX(c, \mathcal{D}_{i,t}, \eta)$ to within an additive $\pm\xi(1-2\eta)$. As above, we can obtain such an estimate of $\tilde{\rho}$ (with probability $1-\delta$) by making at most $O(\frac{T^2 \log 1/\delta}{\tau\xi^2(1-2\eta)^2})$ many draws from $EX(c, \mathcal{D}, \eta)$.

Thus, we may assume that we have an approximation ρ' that is within an additive $\pm\xi$ of ρ. We now use the following lemma (the proof is a routine but tedious error analysis applied to the proof of Lemma 7 of [10]):

Lemma 9. *Consider the rejection sampling procedure defined in Lemma 6 but with ρ' used in place of ρ in the definition of p_r and p_f. Given any $\nu > 0$, there is a value $\xi = \frac{1}{poly(1/\tau, 1/\nu)}$ such that if $|\rho' - \rho| \leq \xi$, then*

(i) this procedure rejects with probability at most $1 - (\eta + \frac{\tau}{4})$ (and thus the rejection sampling procedure incurs a slowdown of at most $\frac{4}{\tau}$);

(ii) if the procedure accepts, then the (x, b) that it accepts is distributed according to $(\frac{1}{2} - \rho)EX(c, (\mathcal{D}_{i,t})^+, \eta_+) + (\frac{1}{2} + \rho)EX(c, (\mathcal{D}_{i,t})^-, \eta_-)$ where $|\rho|$ is at most ν and η_+, η_- each differ from η' by at most an additive $\pm\nu$.

It is easy to check that since $|\rho|$, $|\eta' - \eta_+|$ and $|\eta' - \eta_-|$ are all at most ν, the total variation distance between $(\frac{1}{2} - \rho)EX(c, (\mathcal{D}_{i,t})^+, \eta_+) + (\frac{1}{2} + \rho)EX(c, (\mathcal{D}_{i,t})^-, \eta_-)$ and $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$ is at most 4ν. Since the weak learner makes at most poly$(\frac{1}{\tau}, \frac{1}{\delta'})$ draws to the example oracle to achieve confidence $1 - \delta'$ that it outputs a satisfactory weak hypothesis when run on $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$, if we choose $\nu = \frac{\delta}{\text{poly}(1/\tau, 1/\delta')}$ then we have confidence $1 - \delta$ that its behavior when run with $(\frac{1}{2} - \rho)EX(c, (\mathcal{D}_{i,t})^+, \eta_+) + (\frac{1}{2} + \rho)EX(c, (\mathcal{D}_{i,t})^-, \eta_-)$ will be indistinguishable from its behavior when run with $EX(c, \widehat{\mathcal{D}_{i,t}}, \eta')$. This in turn implies that we may take $\xi =$poly$(\frac{1}{\tau}, \frac{1}{\delta})$, and thus that the overall runtime for each invocation of the weak learner to get $g_{i,t}$ is at most poly$(T, \frac{1}{\tau}, \frac{1}{\delta})$. (Recall that $\tau < \frac{1}{2} - \eta$ so a poly$(\frac{1}{\tau})$ factor subsumes a poly$(\frac{1}{1-2\eta})$ factor.)

Finally, as in Section 6.3, we must balance the hypothesis $g_{i,t}$ to obtain the hypothesis $h_{i,t}$ that will actually be used at node $v_{i,t}$. As in that earlier section, we can only approximately balance the hypothesis but this is sufficient by an analysis similar to that of the previous section. So all in all, we have that in case (3) of `Noise-Tolerant SMartiBoost`, the time (and number of calls to $EX(c, \mathcal{D}, \eta)$ required is poly$(T, \frac{1}{\gamma}, \frac{1}{\tau}, \frac{1}{\delta})$ =poly$(\frac{1}{\gamma}, \frac{1}{\tau}, \frac{1}{\delta})$.

Putting all the pieces together, we have the following theorem which establishes correctness and efficiency of the sampling-based version of `Noise-Tolerant SMartiBoost`:

Theorem 6. *Given any τ such that $\eta + \tau < \frac{1}{2}$, let $T = \Theta(\frac{\log(1/\tau)}{\gamma^2})$. If A is a noise-tolerant weak learning algorithm with advantage γ, then* `Noise-Tolerant SMartiBoost` *makes poly$(\frac{1}{\gamma}, \frac{1}{\tau}, \frac{1}{\delta})$ many calls to $EX(c, \mathcal{D}, \eta)$ and with probability $1 - \delta$ outputs a final hypothesis h that satisfies $\Pr_{x \in \mathcal{D}}[h(x) \neq c(x)] \leq \eta + \tau$.*

8 Conclusion

We are working on implementing the algorithm and evaluating its performance and noise tolerance on real world data.

References

[1] N. Alon, J. Spencer, and P. Erdos. *The Probabilistic Method*. Wiley-Interscience, New York, 1992.

[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] N. Bshouty and D. Gavinsky. On boosting with optimal poly-bounded distributions. *Journal of Machine Learning Research*, 3:483–506, 2002.

[4] F. Chung and L. Lu. Concentration inequalities and martingale inequalities. *Internet Mathematics*, 3(1):79–127, 2006.

[5] 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.

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

[7] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[8] 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.

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

[10] A. Kalai and R. Servedio. Boosting in the presence of noise. In *Proceedings of the 35th Annual Symposium on Theory of Computing (STOC)*, pages 196–205, 2003.

[11] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *AAAI/IAAI*, pages 546–551, 1997.

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

[13] Llew Mason, Peter L. Bartlett, and Jonathan Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 38(3):243–255, 2000.

[14] G. R atsch, T. Onoda, and K.-R. M uller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

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

[16] R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003. Preliminary version in *Proc. COLT'01*.

A Sub-Martingales and Azuma's Inequality for Sub-Martingales

A *sub-martingale* is a sequence $X_0, \ldots, X_T$ of random variables with finite means such that for $1 \leq i \leq T$, we have $\mathbf{E}[X_i | X_{i-1}] \geq X_{i-1}$. (This is the definition given in [1]; the definition in [4] is stronger in that it requires $\mathbf{E}[X_i | X_1, \ldots, X_{i-1}] = X_{i-1}$.)

The following theorem is well known, but for completeness we give a proof since we could not find precisely this statement in the literature. The proof is nearly identical to the proof of Theorem 5.2 (Azuma's inequality for regular martingales) given in [4].

Theorem 7. *[Azuma's Inequality for Sub-Martingales] Let $0 = X_0, \ldots, X_T$ be a sub-martingale which has bounded differences, i.e.*

$$|X_i - X_{i-1}| \leq c_i \quad \textit{for each } i = 1, \ldots, T.$$

Then for any $\lambda > 0$ we have

$$\Pr[X_T \leq -\lambda] \leq \exp\left(-\frac{\lambda^2}{2\sum_{i=1}^T c_i^2}\right).$$

Proof. As in [4], for any fixed θ and $|x| \le c$ we have

$$e^{\theta x} \le \frac{1}{2c}(e^{\theta c} - e^{-\theta c})x + \frac{1}{2}(e^{\theta c} + e^{-\theta c}).$$

For $i = 1, \dots, T$ let $Y_i = X_i - X_{i-1}$, so $|Y_i| \le c_i$ and $\mathbf{E}[Y_i|X_{i-1}] \ge 0$. Then for $\theta < 0$ we have

$$\begin{aligned}
\mathbf{E}[e^{\theta Y_i}|X_{i-1}] &\le \mathbf{E}\left[\frac{1}{2c_i}(e^{\theta c_i} - e^{-\theta c_i})Y_i + \frac{1}{2}(e^{\theta c_i} + e^{-\theta c_i})\big|X_{i-1}\right] \\
&= \frac{1}{2c_i}(e^{\theta c_i} - e^{-\theta c_i})\mathbf{E}[Y_i|X_{i-1}] + \frac{1}{2}(e^{\theta c_i} + e^{-\theta c_i}) \\
&\le \frac{1}{2}(e^{\theta c_i} + e^{-\theta c_i}) \\
&\le e^{\theta^2 c_i^2/2}
\end{aligned}$$

where the last line uses the inequality $\cosh(x) \le e^{x^2/2}$, valid for all $x > 0$.

Since

$$\mathbf{E}[e^{\theta Y_i}|X_{i-1}] = \mathbf{E}[e^{\theta(X_i - X_{i-1})}|X_{i-1}] = \mathbf{E}[e^{\theta X_i}|X_{i-1}]e^{-\theta X_{i-1}},$$

we have

$$\mathbf{E}[e^{\theta X_i}|X_{i-1}] \le e^{\theta^2 c_i^2/2} \cdot e^{\theta X_{i-1}}.$$

So inductively, we have

$$\begin{aligned}
\mathbf{E}[e^{\theta X_T}] &= \mathbf{E}[\mathbf{E}[e^{\theta X_T}|X_{T-1}]] \\
&\le e^{\theta^2 c_T^2/2}\mathbf{E}[e^{\theta X_{T-1}}] \\
&\vdots \\
&\le \left(\prod_{i=1}^{T} e^{\theta^2 c_i^2/2}\right)\mathbf{E}[e^{\theta X_0}] \\
&= e^{\frac{1}{2}\theta^2 \sum_{i=1}^{T} c_i^2}.
\end{aligned}$$

Consequently for $\theta < 0$ we have

$$\begin{aligned}
\Pr[X_T \le -\lambda] &= \Pr[e^{\theta X_T} \ge e^{-\theta\lambda}] \\
&\le e^{\theta\lambda}\mathbf{E}[e^{\theta X_T}] \\
&\le e^{\theta\lambda + \frac{1}{2}\theta^2 \sum_{i=1}^{T} c_i^2}.
\end{aligned}$$

Choose $\theta = -\frac{\lambda}{\sum_{i=1}^{T} c_i^2}$ to minimize this expression, and we have

$$\Pr[X_T \le -\lambda] \le \exp\left(-\frac{\lambda^2}{2\sum_{i=1}^{T} c_i^2}\right).$$

□