Technical Reports on Mathematical and Computing Sciences: TR-C138
title: MadaBoost: A modification of AdaBoost
author: Carlos Domingo and Osamu Watanabe
affiliation:
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
Meguro-ku Ookayama, Tokyo 152-8552.
email: watanabe@is.titech.ac.jp
acknowledgments to financial supports:
Supported in part by the Ministry of Education, Science, Sports and Culture,
Grant-in-Aid for Scientific Research on Priority Areas (Discovery Science), 1999.

Abstract. In the last decade, one of the research topics that has received a great deal of attention from the machine learning and computational learning communities has been the so called boosting techniques. In this paper, we further explore this topic by proposing a new boosting algorithm that mends some of the problems that have been detected in the, so far most successful boosting algorithm, **AdaBoost** due to Freund and Schapire [FS97]. These problems are: (1) **AdaBoost** cannot be used in the boosting by filtering framework, and (2) **AdaBoost** does not seem to be noise resistant. In order to solve them, we propose a new boosting algorithm **MadaBoost** by modifying the weighting system of **AdaBoost**. We first prove that one version of **MadaBoost** is in fact a boosting algorithm. Second, we show how our algorithm can be used and analyzed its performance in detail. Finally, we show that our new boosting algorithm can be casted in the statistical query learning model [Kea93] and thus, it is robust to random classification noise [AL88]. (This is a revised version of TR-C133.)

1 Introduction

In the last decade, "boosting techniques" have been received a great deal of attention from the machine learning and computational learning communities. In this paper, we further explore this topic by proposing a new boosting algorithm — **MadaBoost** — that mends some of the problems that have been detected in the, so far most successful boosting algorithm, **AdaBoost** due to Freund and Schapire [FS97]. These problems are: (1) **AdaBoost** cannot be used in the boosting by filtering framework, and (2) **AdaBoost** does not seem to be noise resistant. The outline of our modification was first proposed in [Wat99] with only a partial proof for its justification. In this paper, we describe the modification in detail, provide a much improved analysis of its correctness and performance, and prove that our new boosting algorithm can be casted in the statistical query learning model [Kea93] and thus, it is robust to random classification noise [AL88] as well as to some other kinds of less benign noise [Dec93].

While the above problem (2) is an obvious weakness, it may not seem so important that even if the boosting algorithm does not work for the "filtering framework", where examples are randomly obtained with respect to a distribution defined over all instance space. Here we argue that it is indeed important that the boosting algorithm can be used in the filtering framework.

Recall that **AdaBoost** is defined for the "subsampling framework", where a sample of sufficient size, which is randomly selected before the boosting, is fixed throughout all the boosting process and distributions are defined only with respect to the sample. From the theoretical side, one consequence of having a boosting algorithm for the filtering framework is that we can directly get a bound on the generalization error of the boosting algorithm. From a more practical side, the advantage becomes more clear. First, we do not need to determine "sufficient sample size" for the boosting process. (Although some formulas are provided to calculate sample size, they may not be easy to use and they usually give overestimated size.) Secondly, since a sample is not a priori fixed, we can run the weak learner on random samples of appropriate sizes at each iteration of the boosting; in this way, we can reduce the computation time particularly when the dataset is very large and we use sampling for scaling-up the weak learner [DGW98]. Recall that **AdaBoost** is defined for the subsampling framework, and in fact, as we show in Section 4, it is not appropriate for the filtering framework, at least not in an obvious way.

Note that there are boosting algorithms, namely, the one proposed by Schapire [Sch90] and the one by Freund [Fre95], that can be used for the filtering framework and that are noise resistant [AD93] in the sense of the statistical query learning model. However, none of known algorithms have been "adaptive" like **AdaBoost** or like the algorithm we propose here, a property that has been repeatedly shown to be crucial for the practical feasibility of a boosting algorithm. More recently, some attempts have been done in order to improve **AdaBoost**, mainly to try to make it noise resistant. These modifications also tried to change the weighting scheme so that the weights change more smoothly (see, for instance [FHT98]). From these variations, the only one that is a boosting algorithm in the PAC sense is the BrownBoost recently proposed by Freund [Fre99]. While BrownBoost seems more error resistant than **AdaBoost**, it has not been shown whether it can be used in the filtering framework. Moreover, BrownBoost is a more complicated than **AdaBoost**, while our algorithm remains as simple as **AdaBoost**.

Our modification for **MadaBoost** is very simple. We just bound the weight assigned to each example by its initial probability. In this way, the weights of the examples cannot become arbitrarily large as it happens in **AdaBoost**. The uncontrolled growth of the weights seems to be the root of the problems of **AdaBoost** and we provide some evidence in this direction by showing that this modified weighting scheme works under the filtering framework and belongs to the statistical query model. What is more important and of theoretical interest is that we could show that even this moderate weight scheme has the boosting property. Unfortunately, our proof still has some minor weaknesses. First, the proof works only for one version of **MadaBoost** that has an even more moderate weight scheme. Secondly, we need to assume that the advantages (its difference from random guessing) of the weak hypotheses produced during the boosting process are monotonically decreasing. Thirdly, the boosting speed, at least what we could prove, is much slower than the one for **AdaBoost**. Nevertheless, we do not expect these weaknesses to affect the practicality of our algorithm. (See the discussions in the later sections.)

This paper is organized as follows. In the following section, we review the basic notions of boosting and algorithm **AdaBoost** in detail, discussing its advantages and disadvantages in

order to clearly establish our goal. In Section 3 we describe **MadaBoost**, our modification of **AdaBoost** that we propose, and for one version of **MadaBoost**, we prove a theorem concerning its efficiency in the subsampling framework. Then, in Section 4 we describe in detail how to use our algorithm for the filtering framework and why **AdaBoost** seems not to work and discuss its generalization error. Finally, in Section 5 we prove that **MadaBoost** belongs to the statistical query learning model. We conclude in Section 6 with some concluding remarks and highlighting future work.

2 Motivation and Our Technical Goal

Here we explain our motivation more specifically and state the technical goal of this paper. (Those who are familiar with the notions of PAC learning, boosting, and **AdaBoost** can skip the following explanation and subsection. But please do not skip Section 2.2, where we give the description of **AdaBoost** in a different way from the original description.)

We begin by recalling some basic notions from computational learning theory, in particular, on PAC learning (see, e.g., [KV94]). The goal of PAC learning is to obtain some hypothesis f that can predict an unknown target function f_* with a desired accuracy under an unknown distribution D on the domain X of f_*. Throughout this paper, we consider the problem of learning binary functions; that is, the range of our target function f_* is $\{0,1\}$. In the PAC learning model, a learning algorithm can generate labeled examples by using some black box example generator $\mathbf{EX}_{D,f_*}$, which generates a pair $(x, f_*(x))$ with probability $D(x)$, and it is expected to obtain some f that is close to f_*. More precisely, an algorithm A is called a *PAC learner* (in the strong sense) if for any D, and for any input ϵ and δ, $0 < \epsilon$, $\delta < 1$, by using $\mathbf{EX}_{D,f_*}$, A yields some hypothesis f such that $\mathrm{cor}_D(f, f_*)$ ($\stackrel{\mathrm{def}}{=} \Pr_{x:D}\{\, f(x) = f_*(x)\,\}$) $\geq 1 - \epsilon$ with probability at least $1 - \delta$. On the other hand, an algorithm yielding a hypothesis that is better than the random hypothesis, i.e., the hypothesis predicting 0 or 1 uniformly at random, is called a weak PAC learner. More precisely, for any D and any input δ, $0 < \delta < 1$, a *weak PAC learner* uses $\mathbf{EX}_{D,f_*}$ and obtains, with probability at least $1 - \delta$, a hypothesis h such that $\mathrm{cor}_D(h, f_*) = 1/2 + \gamma$ for some $\gamma > 0$. This γ is called the *advantage* of h (over the random hypothesis). The efficiency (e.g., the running time) of a learning algorithm is usually measured in terms of $1/\epsilon$, $1/\delta$, the size of instances, and the "complexity" of a target function (resp., for weak learners, $1/\gamma$, $1/\delta$, the size of instances, and the "complexity" of a target function). In order to simplify our discussion, we will ignore instance size and complexity of the target function, and we measure the efficiency only in terms of $1/\delta$ and $1/\epsilon$ and/or $1/\gamma$.

2.1 Boosting Techniques: Subsampling and Filtering

The question of whether a weak PAC learning algorithm is really weaker than a strong PAC learning algorithm was answered negatively by Schapire [Sch90]. He showed that any weak PAC learning algorithm could be transformed into a strong PAC learning algorithm by using what he called "boosting techniques".

We explain the outline of boosting techniques. Suppose that we are given some weak learning algorithm **WeakLearn**. For the rest of the paper we will assume that **WeakLearn** is an algorithm that, when given as input a confidence value δ, it asks for a certain amount of labeled examples under a fixed but unknown distribution and, with probability larger than $1-\delta$, outputs a weak hypothesis that has error strictly smaller than $1/2$ under the distribution from which the examples were generated.

A boosting algorithm runs this **WeakLearn** several times, say T times, under distributions $D_1, ..., D_T$ that are slightly modified from the given distribution D and collects weak hypotheses $h_1, ..., h_T$. (In the following, we call each execution of **WeakLearn** a *boosting round* or *boosting step.*) A final hypothesis is built by combining these weak hypotheses. Here the key idea is to put more weight, when making a new weak hypothesis, to "problematic instances" for which the previous weak hypotheses perform poorly. That is, at the point when $h_1, ..., h_{t-1}$ have been obtained, the boosting algorithm computes a new distribution D_t that puts more weight on those instances that have been misclassified by most of $h_1, ..., h_{t-1}$. Then a new hypothesis h_t produced by **WeakLearn** on this distribution D_t should be strong on those problematic instances, thereby improving the performance of the combined hypothesis built from $h_1, ..., h_t$.

Boosting techniques differ typically on (i) the weighting scheme used to obtained the modified distribution, (ii) the way the weak hypotheses are combined, and (iii) the way to execute **WeakLearn** on the modified distributions. According to this last point, boosting techniques can be classified in two types:- *boosting by subsampling* and *boosting by filtering* [Fre95] [1].

In the boosting by subsampling framework, a boosting algorithm first obtains, using $\mathbf{EX}_{D,f_*}$, a set S of enough number of examples as a "training set". Then it runs **WeakLearn** on S by changing the weight of each example. The goal is to obtain a hypothesis that explains the training set well and hope that this hypothesis will also predict well the label on unseen examples outside S. Since we can now regard S as a domain, we only need to provide labeled examples under a given distribution on S, which is possible so long as the distribution is efficiently computable. That is, the boosting algorithm can provide examples to **WeakLearn** according to its desired distribution. Or, alternatively, we can modify **WeakLearn** so that it can handle weighted examples. Usually, the initial distribution is set to the uniform distribution on the training set S (and zero everywhere else).

On the other hand, in the boosting by filtering framework, a boosting algorithm selects an example from the original domain X each time **WeakLearn** requests one. This selection procedure is regarded as a "filter" between $\mathbf{EX}_{D,f_*}$ and **WeakLearn**. That is, the filter observes each example generated by $\mathbf{EX}_{D,f_*}$ and either "rejects" it and throws it away or "accepts" it and passes it on to **WeakLearn**; by this process, the boosting algorithm runs **WeakLearn** on modified distributions. Notice that, in this framework, the distributions are defined with respect to the original domain rather than with respect to a sample set as in the subsampling framework. In this case, the initial distribution is the unknown distribution D from which the

[1] These frameworks are also sometimes refer as *boosting by reweighting* and *boosting by resampling* [FS96].

examples are obtained.

2.2 Adaptive Boosting: AdaBoost

Among all the theoretically provable boosting techniques, the most successful one in practical applications has been **AdaBoost** due to Freund and Schapire [FS97]. The explanation of its success comes from two reasons, first its simplicity and second a property of AdaBoost that previous boosting algorithms [Sch90, Fre95] lacked of, namely, "adaptivity". The algorithm adapts its strategy to the situation being used, which free its user from the difficulty of determining algorithmic parameters.

Let us see what the "adaptivity" of **AdaBoost** means. For using any of the previous boosting algorithms [Sch90, Fre95], we need to specify a lower bound γ of the advantage of weak hypotheses that one can expect from **WeakLearn**. This γ, with other parameters, determines an actual boosting strategy (i.e., a weighting scheme and a form of the combined hypothesis), which works well so long as the weak learner keeps producing weak hypotheses with advantage $\geq \gamma$. Furthermore, the total running of the boosting algorithm increases depending on $1/\gamma$, roughly $\mathcal{O}(1/\gamma^2)$. Thus, when using one of these boosting algorithms, one has to estimate γ appropriately; if γ is overestimated, then the boosting process may not work, but if γ is underestimated, the boosting process becomes unnecessarily slow. **AdaBoost** solves this problem! It *adapts* to the advantages of the obtained weak hypotheses and depending on this information, it chooses an *appropriate* strategy *on-line*. Thus, we do not have to estimate γ for using **AdaBoost**.

Now we describe **AdaBoost** in more detail. **AdaBoost** is designed for the subsampling framework. Let S be the sample that is given as input to **AdaBoost**. For any $t \geq 1$, assume that we have already obtained hypotheses $h_1, ..., h_{t-1}$, where each h_i is a weak hypothesis of f_* on some distribution D_{i-1} defined with respect to sample S. Let $\epsilon_1, ..., \epsilon_{t-1}$ denote the errors and $\gamma_1, ..., \gamma_{t-1}$ denote the advantages of these hypotheses; that is, $\epsilon_i \stackrel{\text{def}}{=} 1 - \mathrm{cor}_{D_i}(h_i)$ and $\gamma_i \stackrel{\text{def}}{=} \mathrm{adv}_{D_i}(h_i)$ $(= 1/2 - \epsilon_i)$. We use parameters $\beta_1, ..., \beta_{t-1}$ that are defined as $\beta_i \stackrel{\text{def}}{=} \sqrt{\epsilon_i/(1-\epsilon_i)}$ $(= \sqrt{(1-2\gamma_i)/(1+2\gamma_i)})$ for each i, $1 \leq i \leq t-1$. (*Remark.* In this paper, we define β_i in this way, which is the square root of β_i used in [FS97]. Due to this change, the weight w_t defined below is not the same as the one used in [FS97] although the definitions of the distributions D_t and the combined hypothesis f_t also given below are mathematically equivalent. Thus, there is no essential difference between this description of the algorithm and the original one, while ours is more appropriate for discussing the modification to the filtering framework.) For any hypothesis h and any $x \in X$, define $\mathrm{cons}(h, x) \stackrel{\text{def}}{=} 1$ (resp., -1) if $h(x) = f_*(x)$ (resp., $h(x) \neq f_*(x)$). Then for each instance $x \in S$, its weight $w_{t-1}(x)$ after the $(t-1)$th boosting round is defined as follows.

$$w_{t-1}(x) \stackrel{\text{def}}{=} D_0(x) \times \prod_{1 \leq i \leq t-1} \beta_i^{\mathrm{cons}(h_i, x)},$$

where D_0 is the initial distribution, that is the uniform distribution on S. Let $W_{t-1} \stackrel{\text{def}}{=}$

$\sum_{x \in S} w_{t-1}(x)$. The next distribution D_t is defined as $D_t(x) \stackrel{\text{def}}{=} w_{t-1}(x)/W_{t-1}$ for all $x \in S$. (Note that $D_1(x) = D_0(x)$.) Finally, the combined hypothesis f_{t-1} of $h_1, ..., h_{t-1}$ is their weighted majority vote that is defined as

$$f_{t-1}(x) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } \prod_{i:h_i(x)=1} \beta_i \leq \prod_{i:h_i(x)=0} \beta_i, \\ 0, & \text{otherwise.} \end{cases}$$

Then for the "boosting property" of **AdaBoost**, we have the following theorem proved by Freund and Schapire in [FS97].

Theorem 1 *Suppose that* **WeakLearn***, when called by* **AdaBoost** *on a training set S, generates weak hypotheses $h_1, h_2, ..., h_T$ whose advantages are $\gamma_1, \gamma_2, ..., \gamma_T$. Then the error of the combined hypothesis f_T on S can be bounded as follows.*

$$\text{error}(S, f_T) \stackrel{\text{def}}{=} \Pr_{x:D_0} \{ f_T(x) \neq f_*(x) \} \leq \exp\left(-2 \sum_{i=1}^{T} \gamma_i^2\right).$$

Now we calculate the number of boosting rounds T needed in order to make f_T consistent with the sample S. Suppose as in the theorem that **WeakLearn** produces hypotheses whose advantages are $\gamma_1, \gamma_2, ..., \gamma_T$. Then it follows from the theorem that if $\sum_{i=1}^{T} \gamma_i^2 \geq (\ln \|S\|)/2$, then we have that $\text{error}(S, f_T) < 1/\|S\|$, which means that f_T does not make any error on S. Hence, for any lower bound γ of $\gamma_1, ..., \gamma_T$, we have $T = \mathcal{O}(1/\gamma^2)$. That is, even if we do not supply lower bound γ, **AdaBoost** takes an appropriate strategy and performs at least as well as the previous boosting techniques with the best parameter γ.

2.3 Drawbacks of AdaBoost and Our Technical Goal

Although its indisputable success, **AdaBoost** has two main problems that we address in this paper. Solving them while keeping the good properties of **AdaBoost**, namely, simplicity and adaptivity, is our main objective. Let us first describe what are these two problems.

Recall that **AdaBoost** is designed for the subsampling framework. The subsampling framework has a number of disadvantages, both from a theoretical and a practical points of view, which can be avoided by having a boosting algorithm that works for the filtering framework. In the following, we discuss the advantages of having a version of **AdaBoost** for the filtering framework.

One of the main theoretical drawbacks of **AdaBoost** is the lack of a tight bound on its generalization error. In Theorem 1, we have seen that, assuming the existence of a **WeakLearn**, **AdaBoost** produces a hypothesis that is consistent with a given set of examples; that is, the training error can be driven down to zero. However, we are usually interested in how a hypothesis performs on unseen instances; that is, we want to bound the generalization error. One way to do this could be by restricting the weak learner to choose its hypothesis from a simple class of functions and then restrict the number of hypotheses that are going to be

combined, i.e., the number of boosting steps. Then using uniform convergence theory we can bound the generalization error of **AdaBoost** [FS97]. In this way we can upper bound the generalization error of **AdaBoost** by $\widehat{\epsilon} + \mathcal{O}(\sqrt{Td/|S|})$, where $\widehat{\epsilon}$ is the error in the training set S, T is the number of boosting rounds, and d is the VC dimension of the **WeakLearn** hypothesis space. It has been observed, however, by several researchers [DC96, FS96, Bre98] that this bound does not reflect the actual behavior of **AdaBoost**. Even after the training error has reached zero, **AdaBoost** sometimes continues to drive down the generalization error contradicting the bound. A seemingly better bound in terms of a quantity called the margin has been proposed by Schapire et.al. [SFBL98]. While this bound seems qualitatively right it is still greatly underestimating the actual performance of **AdaBoost** and thus not satisfactory. From the statistical community, Breiman [Bre98] has proposed an explanation in terms of bias-variance decomposition but without providing a provable theorem about the generalization error of **AdaBoost**.

Notice that a version of **AdaBoost** designed for the filtering framework necessarily provides a direct bound on the generalization error as it was done with previous boosting algorithms [Sch90, Fre95]. Recall that in the filtering framework a new training sample is obtained (randomly with respect to the current distribution) at each boosting iteration from which a new weak hypothesis is constructed. Thus, the "boosting property" of a boosting algorithm for the filtering framework should be proved directly on the generalization error.

Other advantage of the boosting by filtering framework is that the task of determining the appropriate sample size does not have to be done a priory. In fact, we could use a different sample size at each boosting iteration, and this point could be used for improving total efficiency. For example, suppose that we have a **WeakLearn** that observes examples under a certain distribution and determines in an on-line manner when it has seen enough examples to output a hypothesis [DGW98]. This sort of **WeakLearn** can be extremely efficient in minimizing the number of examples used compared to the usual **WeakLearn** where the total number of examples needed is usually determined a priori by using a worst case bound that works for any distribution. Moreover, this could be particularly useful in practical situations where huge amount of data is available but we do not need to use it all at each step to construct a weak hypothesis. Notice that a **WeakLearn** like this can be boosted only under the filtering framework and thus, **AdaBoost** (used in the subsampling framework) is not appropriate for it.

Unfortunately, though, **AdaBoost** does not seem suitable for the boosting by filtering framework, at least in a straightforward way. If one constructs the filter in the obvious way (i.e., obtaining one example under the original distribution and then using its weight to determine the probability that the example is passed or not to the **WeakLearn**), then after certain boosting iterations, the probability that the filter outputs any example may become exponentially smaller than the current error of the combined hypothesis. Thus, while we still need to do some more boosting iterations to reach the desired error bound, we cannot obtain enough examples from the filter for **WeakLearn** to work. The technical problem is that after certain number of boosting iterations, the weights of the instances can grow exponentially. Thus, most of the weight under

the current probability distribution is placed in a fraction with very small weight under the original probability distribution, and hence, most of the examples are rejected by the filter. This problem is described in more detail in Section 4.

The second main problem is that **AdaBoost** does not seem to be noise resistant. A recent empirical study done by Dietterich [Die98] shows that the performance of **AdaBoost** in noisy domains gets degradated. The reason seems to be that more and more weight is being placed on the noisy examples forcing the weak learner to agree with them; thus, the weak learner obtains erroneous hypotheses that degradate the generalization error of the combined hypothesis. While this problem seems difficult to solve in general, we may still be able to treat noisy data of some type. For example, if a learning algorithm can be used for the statistical query learning of Kearns [Kea93], we can handle random classification noise [AL88] and several other kinds of less benign noise [Dec93]. In fact, Aslam and Decatur [AD93] showed that previous boosting algorithms have this property. On the other hand, it seems difficult to show it for **AdaBoost** due to again its weighting scheme that may put very large weight to some instances and the fact that it is not designed for the filtering framework (the proof in [AD93] heavily relies in this fact).

Now from the discussion above, our technical goal becomes clear. We would like to modify the weighting scheme of **AdaBoost** so that no instance gets an extremely large weight. With such a weighting scheme the filtering algorithm can produce each example in a reasonable amount of time, and thus, we could use this new boosting algorithm in the filtering framework. From this, a bound on the generalization error and noise resistant properties should follow. Moreover, we want do this modification while keeping the good properties of AdaBoost, in particular, the adaptive boosting property.

3 MadaBoost: A New Weighting Scheme

In the previous section, we stated our technical goal. That is, while keeping the adaptive boosting property, we want to modify the weighting scheme of **AdaBoost** so that (i) it can be used in the filtering framework, and (ii) it can be used for the statistical query learning. Here we describe our modification for our boosting algorithm **MadaBoost** and prove that one version of **MadaBoost** has an adaptive boosting property.

Since **AdaBoost** is designed for the subsampling framework, for the sake of comparison we will also describe **MadaBoost** in the subsampling framework. In Section 4 we will explain how to make it work for the filtering framework.

The modification we propose is very simple. For each instance x, we modify its weight $w_t(x)$ after the tth boosting round with the initial weight $D_0(x)$ as a saturation bound; that is, the weight $w_t(x)$ cannot be increased beyond $D_0(x)$. (Recall that D_0, in the subsampling framework, is usually taken as the uniform distribution over S.) More specifically, our new weighting system is defined as follows. For each instance $x \in S$, the new weight $w_{t-1}(x)$ after the $(t-1)$th step is defined by

$$w_{t-1}(x) \stackrel{\text{def}}{=} \begin{cases} D_0(x) \times \prod_{1 \le i \le t-1} \beta_i^{\text{cons}(h_i,x)}, & \text{if } \prod_{1 \le i \le t-1} \beta_i^{\text{cons}(h_i,x)} < 1, \text{ and} \\ D_0(x), & \text{otherwise.} \end{cases}$$

The rest is exactly the same as before. That is, we define $W_{t-1} \stackrel{\text{def}}{=} \sum_{x \in S} w_{t-1}(x)$, and the next distribution D_t is defined by $D_t(x) \stackrel{\text{def}}{=} w_{t-1}(x)/W_{t-1}$ for all $x \in S$, and also the combined hypothesis is exactly the same as for **AdaBoost** (see Section 2.2).

Thus, the weight of each instance changes very moderately. It is interesting to see that even using this moderate weighting scheme the algorithm still has some boosting property. For example, our experiments [DW99b] show that **MadaBoost** has a boosting property more or less similar to **AdaBoost**. We can also prove some basic boosting property of **MadaBoost** for a special case where (we may assume that) the advantage of every obtained hypothesis is some fixed $\gamma > 0$ [Wat99].

Here we prove a more general boosting property. Unfortunately, though, for our current proof, we need to modify the weighting scheme of **MadaBoost** even more moderate one. The difference from **MadaBoost** is the definition of β_t; here the following definition is used.

$$\beta_t \;=\; \sqrt{\frac{\epsilon'_t}{1-\epsilon'_t}}, \qquad \text{where } \epsilon'_t \;=\; \frac{\sqrt{\epsilon_t}}{\sqrt{2}}.$$

That is, instead of using the error probability ϵ_t of the tth weak hypothesis, we use ϵ'_t, which is slightly larger than ϵ_t. In terms of the advantage γ_t, we have

$$\epsilon'_t \;=\; \frac{\sqrt{\epsilon_t}}{\sqrt{2}} \;=\; \frac{\sqrt{1-2\gamma_t}}{2} \;\approx\; \frac{1-\gamma_t}{2} \;=\; \frac{1}{2} - \frac{\gamma_t}{2}.$$

Hence, roughly speaking, this new weighting scheme defines β_t by using the advantage that is the half of the real one. Note also that $\epsilon'_t < 1/2$ so long as $\epsilon_t < 1/2$. We refer this version of **MadaBoost** as **MB:1/2**.

For this **MB:1/2**, we can prove the following boosting property.

Theorem 2 *Suppose that WLearn, when called by* **MB:1/2** *on a training set S, generates weak hypotheses $h_1, h_2, ..., h_T$ whose advantages are $\gamma_1, \gamma_2, ..., \gamma_T$ that satisfies $\gamma_1 \ge \gamma_2 \ge \cdots \ge \gamma_T > 0$. Let $f_1, f_2, ..., f_T$ be combined hypotheses obtained after each round. Then for any ϵ, either there is some t, $1 \le t < T$, for which we have*

$$\text{error}(S, f_t) \;=\; \Pr_{x:D_0}\{\, f_t(x) \neq f_*(x) \,\} \;<\; \epsilon,$$

or we have the following bound.

$$\text{error}(S, f_T) \;=\; \Pr_{x:D_0}\{\, f_T(x) \neq f_*(x) \,\} \;\le\; 1 - \sum_{i=1}^{T} 2\epsilon\gamma_i^2.$$

Before the proof, let us examine the meaning of this theorem. Suppose as before that our goal is to obtain a hypothesis whose error probability on S is less than $\epsilon = 1/\|S\|$ (under the uniform distribution over S), and suppose that **WeakLearn** generates weak hypotheses with advantage larger than γ. Then within $T = (\|S\| - 1)/(2\gamma^2)$ rounds, we have some f_t with $\mathrm{error}(S, f_t) < \epsilon$ because if no f_t, $1 \leq t < T$, has this desired property, then we have

$$\mathrm{error}(S, f_T) \;\leq\; 1 - \frac{\|S\| - 1}{2\gamma^2} \cdot \frac{2\gamma^2}{\|S\|} \;\leq\; \frac{1}{\|S\|},$$

that is, f_T indeed has the desired property. Hence, we can bound the boosting rounds as $\mathcal{O}(\|S\|/\gamma^2)$, or $\mathcal{O}(\epsilon^{-1}/\gamma^2)$ in general to reduce the error in S below ϵ. Recall, on the other hand, that **AdaBoost**, as well as many other boosting techniques, needs $\mathcal{O}(\ln \epsilon^{-1}/\gamma^2)$ under the same situation. That is, the boosting speed of **MB:1/2**, at least the one we can prove here, is exponentially slower than **AdaBoost** in terms of ϵ^{-1}.

Here we need one condition; that is, the advantage sequence $\gamma_1, \gamma_2, \ldots, \gamma_T$ is non-increasing. Intuitively, a learning problem gets harder and harder as boosting proceeds; hence, it seems natural to assume that an advantage sequence is non-increasing. In fact, this phenomenon has been confirmed experimentally by Dietterich et.al. in dietterich-kearns-mansour-icml96, where it was found that the distributions generated by **AdaBoost** are increasingly difficult for the **WeakLearn** and thus, the advantage sequence is monotonically decreasing. Similar results have been also obtained by our experiments [DW99b] for **MadaBoost** and **MB:1/2**.

Note that, even if it occurs $\gamma_t > \gamma_{t-1}$ for some round t, we can continue boosting by using γ_{t-1} instead of γ_t. That is, **MB:1/2** works with no problem with any advantage sequence; it just cannot use the advantage of some "accidentally" good weak hypothesis.

For proving the theorem, we first note the following fact, which will be also important for our discussion in the next section.

Lemma 3 *Suppose that* **MB:1/2** *is executed with a sample set S in which it executes* **WeakLearn** *for t times and obtains weak hypotheses $h_1, ..., h_t$ with advantage $\gamma_1, ..., \gamma_t$. Let f_t be the combined hypothesis obtained from them. Then we have*

$$\mathrm{error}(S, f_t) \;=\; \Pr_{x:D_0}\{\, f_t(x) \neq f_*(x) \,\} \;\leq\; W_t.$$

Remark. *The proof follows from the definition of f_t and W_t, and it is omitted here. Note that the same proof works independent from the choice of β_t. Hence, the theorem holds for* **MadaBoost**. *Furthermore, the theorem also holds even for* **AdaBoost** *(under the weigting scheme as described in Section 2.)*

Therefore, in order to prove Theorem 2, it suffices to show that the sequence $W_1, W_2, ...$ converges to 0. More specifically, we need to show that W_t is smaller than W_{t-1} by $(\epsilon/2)\gamma_t^2$ (if $W_{t-1} \geq \epsilon$). This is what we will prove below.

Proof of Theorem 2. For any $t \geq 1$, we consider the situation when the tth boosting round has just finished. (Below the ith boosting round is simply called "the ith step".) That is,

WeakLearn has been called for t times and t weak hypotheses $h_1, \ldots, h_t$ have been obtained. Let $\epsilon_1, \ldots, \epsilon_t$ be their errors, and let $\gamma_1, \ldots, \gamma_t$ be their advantages. We would like to discuss how much W_t gets decreased from W_{t-1}, but it seems difficult to estimate the decrement because there may be some instance $x \in X$ for which the weight does not change between the $(t-1)$th and the tth step. Thus, we introduce here some "imaginary" weight that bounds W_t and show that it gets decreased as the theorem claims.

First we define our new weight for the situation that the $(t-1)$th step has been finished. For this, we divide the instance space X according to the value $\prod_{i=1}^{t-1} \beta_i^{\mathrm{cons}(h_i,x)}$ of each instance $x \in X$. (For simplifying our notation, we use below $B_{t-1}(x) \stackrel{\mathrm{def}}{=} \prod_{i=1}^{t-1} \beta_i^{\mathrm{cons}(h_i,x)}$.) More specifically, we divide X into two sets U and V, where U (resp., V) is the set of instances $x \in X$ such that $B_{t-1}(x) < 1$ (resp., $B_{t-1}(x) \geq 1$). Note that x is in U iff f_{t-1} gives a correct classification, and note that for any $x \in V$, its weight $w_{t-1}(x)$ is defined by $w_{t-1}(x) = D_0(x)$. Now we define our new weight $\widetilde{w}_{t-1}(x)$ as follows.

$$\begin{array}{lrcl} (\text{if } x \in U) & \widetilde{w}_{t-1}(x) & \stackrel{\mathrm{def}}{=} & B_{t-1}(x) D_0(x), \quad \text{and} \\ (\text{if } x \in V) & \widetilde{w}_{t-1}(x) & \stackrel{\mathrm{def}}{=} & (1 + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x)) D_0(x). \end{array}$$

Where $\alpha_t \stackrel{\mathrm{def}}{=} \beta_t^{-1} - 1 > 0$. Also define $\widetilde{W}_{t-1} \stackrel{\mathrm{def}}{=} \sum_{x \in X} \widetilde{w}_{t-1}(x)$.

We explain the motivation of this weight function. In our original weight scheme, all instances in U changes their weight from the $(t-1)$th step to the tth step, while some instances x in V keep the same weight, i.e., $D_0(x)$. Thus, for the weight $\widetilde{w}_{t-1}(x)$ for each $x \in V$, we introduce the additional *nonnegative* term $\alpha_t \log_{\beta_t^{-1}} B_{t-1}(x) D_0(x)$ so that $\widetilde{w}_{t-1}(x)$ changes depending whether h_t gives the correct classification of x or not. Intuitively, if h_t correctly (resp., wrongly) classifies $x \in V$, then its weight gets decreased (resp., increased) by α_t.

We need to be a bit careful for defining the new weight for the tth step. The weight $\widetilde{w}_t$ after the tth step is defined similarly, but the division of X is different and β_{t+1} should be used here. That is, these are defined as follows by using $B_t(x) \stackrel{\mathrm{def}}{=} \prod_{i=1}^{t} \beta_i^{\mathrm{cons}(h_i,x)}$ and $\alpha_{t+1} \stackrel{\mathrm{def}}{=} \beta_{t+1}^{-1} - 1$.

$$\begin{array}{lrcl} U' & \stackrel{\mathrm{def}}{=} & \{\, x \in X \mid B_t(x) < 1 \,\}, \\ V' & \stackrel{\mathrm{def}}{=} & \{\, x \in X \mid B_t(x) \geq 1 \,\}, \\ (\text{if } x \in U') & \widetilde{w}_t(x) & \stackrel{\mathrm{def}}{=} & B_t(x) D_0(x), \quad \text{and} \\ (\text{if } x \in V') & \widetilde{w}_t(x) & \stackrel{\mathrm{def}}{=} & (1 + \alpha_{t+1} \log_{\beta_{t+1}^{-1}} B_t(x)) D_0(x). \end{array}$$

Then define $\widetilde{W}_t \stackrel{\mathrm{def}}{=} \sum_{x \in X} \widetilde{w}_t(x)$. On the other hand, for our comparison, we also consider an intermediate weight function $\widetilde{w}'_t$ that is defined as follows.

$$\begin{array}{lrcl} (\text{if } x \in U') & \widetilde{w}'_t(x) & \stackrel{\mathrm{def}}{=} & B_t(x) D_0(x), \\ (\text{if } x \in V') & \widetilde{w}'_t(x) & \stackrel{\mathrm{def}}{=} & (1 + \alpha_t \log_{\beta_t^{-1}} B_t(x)) D_0(x). \end{array}$$

Define $\widetilde{W}'_t \stackrel{\mathrm{def}}{=} \sum_{x \in X} \widetilde{w}'_t(x)$.

Below we will show that $\widetilde{W}_t$ gets decreased from $\widetilde{W}_{t-1}$ by $(\gamma_t^2/2) W_{t-1}$ by proving that (i) $\widetilde{W}_t \leq \widetilde{W}'_t$, and (ii) $\widetilde{W}'_t \leq \widetilde{W}_{t-1} - (\gamma_t^2/2) W_{t-1}$.

First prove $\widetilde{W}_t \le \widetilde{W}'_t$. For this, it suffices to show that $\widetilde{w}_t(x) \le \widetilde{w}'_t(x)$ for all $x \in X$. The case that $x \in U'$ is trivial because by definition $\widetilde{w}_t(x) = \widetilde{w}'_t(x)$. Consider the case $x \in V'$. Here we note that $\beta_{t+1}^{-1} \le \beta_t^{-1}$; this is because we assumed that $\gamma_{t+1} \le \gamma_t$. Thus we have

$$\begin{aligned}
&\widetilde{w}_t(x) \;\le\; \widetilde{w}'_t(x) \\
&\Leftrightarrow\; \alpha_{t+1} \log_{\beta_{t+1}^{-1}} B_t(x) \;\le\; \alpha_t \log_{\beta_t^{-1}} B_t(x) \\
&\Leftrightarrow\; (\beta_t^{-1} - 1)(\log_2 B_t(x) / \log_2 \beta_t^{-1}) \;\le\; (\beta_{t+1}^{-1} - 1)(\log_2 B_t(x) / \log_2 \beta_{t+1}^{-1}) \\
&\Leftrightarrow\; (\beta_t^{-1} - 1) / \log_2 \beta_t^{-1} \;\le\; (\beta_{t+1}^{-1} - 1) / \log_2 \beta_{t+1}^{-1}.
\end{aligned}$$

This last inequality holds because $\phi(z) = (z-1)/\log_2 z$ is monotone for $z > 1$.

Next we analyze how much does $\widetilde{W}'_t$ get decreased from $\widetilde{W}_{t-1}$. For this, we estimate $\Delta(x) = \widetilde{w}'_t(x) - \widetilde{w}_{t-1}(x)$ considering the cases (U) $x \in U$ and (V) $x \in V$. Furthermore, for each case, we consider the case (P) h_t correctly classifies x and the case (Q) h_t wrongly classifies x. Below we use P and Q to denote $\{\, x \in X \mid h_t(x) = f_*(x) \,\}$ and $\{\, x \in X \mid h_t(x) \ne f_*(x) \,\}$ respectively.

(Case U.P) Note that $x \in U'$. Hence, by definition, we have

$$\begin{aligned}
\widetilde{w}_{t-1}(x) &= B_{t-1}(x) D_0(x) \;=\; w_{t-1}(x), \;\text{ and} \\
\widetilde{w}'_t(x) &= B_t(x) D_0(x) \;=\; \beta_t w_{t-1}(x).
\end{aligned}$$

Thus $\Delta(x) = (\beta_t - 1) w_{t-1}(x)$ (≤ 0).

(Case U.Q) We have either $x \in U'$ or $x \in V'$. For the former case, we have

$$\begin{aligned}
\widetilde{w}_{t-1}(x) &= B_{t-1}(x) D_0(x) \;=\; w_{t-1}(x), \;\text{ and} \\
\widetilde{w}'_t(x) &= B_t(x) D_0(x) \;=\; \beta_t^{-1} w_{t-1}(x),
\end{aligned}$$

and hence $\Delta(x) = (\beta_t^{-1} - 1) w_{t-1}(x)$ (≥ 0). For the latter case, we have

$$\begin{aligned}
\widetilde{w}_{t-1}(x) &= B_{t-1}(x) D_0(x), \;\text{ and} \\
\widetilde{w}'_t(x) &= (1 + \alpha_t \log_{\beta_t^{-1}} B_t(x)) D_0(x) \;=\; (1 + \alpha_t + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x)) D_0(x).
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\Delta(x) &= (1 - B_{t-1}(x) + \alpha_t + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x)) D_0(x) \\
&= (\alpha_t + (1 - B_{t-1}(x) + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x))) D_0(x).
\end{aligned}$$

Here we want to show that $1 - B_{t-1}(x) + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x) \le 0$. Note that $\beta_t \le B_{t-1}(x) < 1$ because $x \in U$ (i.e., $B_{t-1}(x) < 1$) and $x \in V'$ (i.e., $B_t(x) = B_{t-1}(x)\beta_t^{-1} \ge 1$). Hence, $B_{t-1}(x) = \beta_t^z$ for some z, $0 < z \le 1$. Then we have

$$1 - B_{t-1}(x) + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x) \;=\; 1 - \beta_t^z - \alpha_t z$$

On the other hand, the function $\psi(z) \stackrel{\text{def}}{=} 1 - \beta_t^z - \alpha_t z$ is nonincreasing on $[0,1]$ (since $\alpha_t = \beta_t^{-1} - 1 \ge -\ln \beta_t$) and $\psi(z) = 0$. Thus, we can show that

$$\Delta(x) \;\le\; \alpha_t D_0(x) \;\le\; \beta_t^{-1} \alpha_t B_{t-1}(x) D_0(x) \;=\; \beta_t^{-1} \alpha_t w_{t-1}(x).$$

(Case V.P) We have either $x \in U'$ or $x \in V'$. For the latter case, we have

$$\begin{aligned} \widetilde{w}_{t-1}(x) &= (1 + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x))D_0(x), \text{ and} \\ \widetilde{w}'_t(x) &= (1 + \alpha_t \log_{\beta_t^{-1}} B_t(x))D_0(x) \;=\; (1 - \alpha_t + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x))D_0(x). \end{aligned}$$

Hence $\Delta(x) = -\alpha_t D_0(x) = -\alpha_t w_{t-1}(x)$ (≤ 0). For the former case, we have

$$\begin{aligned} \widetilde{w}_{t-1}(x) &= (1 + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x))D_0(x), \text{ and} \\ \widetilde{w}'_t(x) &= B_t(x)D_0(x) \;=\; B_{t-1}(x)\beta_t D_0(x) \end{aligned}$$

Let us consider here backwards. For our later analysis, we would like to have $\Delta(x) \leq (\beta_t - 1)w_{t-1}(x)$. But for this, it is enough to have $\Delta(x) \leq (\beta_t - 1)\widetilde{w}_{t-1}(x)$ since $B_{t-1}(x) \geq 1$ and thus $\widetilde{w}_{t-1}(x) \geq w_{t-1}(x)$. On the other hand, since

$$\Delta(x) \;=\; (B_{t-1}(x)\beta_t - (1 + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x)))D_0(x),$$

we have

$$\begin{aligned} &\Delta(x) \;\leq\; (\beta_t - 1)\widetilde{w}_{t-1}(x) \\ &\Leftrightarrow\; B_{t-1}(x)\beta_t - (1 + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x)) \;\leq\; (\beta_t - 1)(1 + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x)) \\ &\Leftrightarrow\; B_{t-1}(x) \;\leq\; 1 + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x) \\ &\Leftrightarrow\; \frac{B_{t-1}(x) - 1}{\log_{\beta_t^{-1}} B_{t-1}(x)} \;\leq\; \alpha_t. \end{aligned}$$

To prove the last inequality, we note that $1 \leq B_{t-1}(x) < \beta_t^{-1}$. Thus, by letting $\beta_t^{-1} = 1 + \delta$ and $B_{t-1}(x) = (1 + \delta)^z$ for some $\delta, z > 0$, we have

$$\frac{B_{t-1}(x) - 1}{\log_{\beta_t^{-1}} B_{t-1}(x)} \;=\; \frac{(1 + \delta)^z - 1}{z} \;\leq\; \frac{z\delta}{z} \;=\; \delta \;=\; \beta_t^{-1} - 1.$$

Thus, the desired inequality holds since we defined $\alpha_t = \beta_t^{-1} - 1$. In fact, this is one of the reasons for our definition of α_t.

(Case V.Q) Since $x \in V'$, we have

$$\begin{aligned} \widetilde{w}_{t-1}(x) &= (1 + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x))D_0(x), \text{ and} \\ \widetilde{w}'_t(x) &= (1 + \alpha_t \log_{\beta_t^{-1}} B_t(x))D_0(x) \;=\; (1 + \alpha_t + \alpha_t \log_{\beta_t^{-1}} B_{t-1}(x))D_0(x). \end{aligned}$$

Hence $\Delta(x) = \alpha_t D_0(x) = (\beta_t^{-1} - 1)w_{t-1}(x)$ (≥ 0).

Summarize our analysis. Note that $\alpha_t = \beta_t^{-1} - 1 \geq 1 - \beta_t$. Then for any $x \in X$, we have shown that

$$\begin{aligned} x \in P \;&\Rightarrow\; \Delta(x) \;\leq\; \max(\beta_t - 1, -\alpha_t)w_{t-1}(x) \;\leq\; (\beta_t - 1)w_{t-1}(x), \text{ and} \\ x \in Q \;&\Rightarrow\; \Delta(x) \;\leq\; \max(\beta_t^{-1}\alpha_t, \beta_t^{-1} - 1)w_{t-1}(x) \;\leq\; \beta_t^{-1}(\beta_t^{-1} - 1)w_{t-1}(x). \end{aligned}$$

Thus, the total difference $\Delta \stackrel{\text{def}}{=} \widetilde{W}'_t - \widetilde{W}_{t-1}$ is estimated as follows.

$$
\begin{aligned}
\Delta &= \sum_{x\in X}\Delta(x) \;=\; \sum_{x\in P}\Delta(x)+\sum_{x\in Q}\Delta(x)\\
&\le \sum_{x\in P}(\beta_t-1)w_{t-1}(x)+\sum_{x\in Q}\beta_t^{-1}(\beta_t^{-1}-1)w_{t-1}(x)\\
&= W_{t-1}\times\left(\sum_{x\in P}(\beta_t-1)D_{t-1}(x)+\sum_{x\in Q}\beta_t^{-1}(\beta_t^{-1}-1)D_{t-1}(x)\right)\\
&= W_{t-1}\times((\beta_t-1)(1-\epsilon_t)+\beta_t^{-1}(\beta_t^{-1}-1)\epsilon_t).
\end{aligned}
$$

Where the last equality is from the definition of P, Q, and ϵ_t; that is, $\epsilon_t = \Pr_{x:D_{t-1}}\{\, h_t(x)\neq f_*(x)\,\} = D_{t-1}(Q)$, and $1-\epsilon_t = \Pr_{x:D_{t-1}}\{\, h_t(x)= f_*(x)\,\} = D_{t-1}(P)$.

Here recall that β_t is defined as $\sqrt{\epsilon'_t/(1-\epsilon'_t)}$ with $\epsilon'_t=\sqrt{\epsilon_t/2}$. Hence we have $\epsilon_t\le\epsilon'_t$ and

$$
\beta_t^{-1}\epsilon_t \;=\; 2(\epsilon'_t)^2\sqrt{\frac{1-\epsilon'_t}{\epsilon'_t}} \;=\; \epsilon'_t(2\sqrt{\epsilon'_t(1-\epsilon'_t)}) \;\le\; \epsilon'_t.
$$

Then we can bound Δ as follows.

$$
\begin{aligned}
\Delta &= W_{t-1}\times((\beta_t-1)(1-\epsilon_t)+\beta_t^{-1}(\beta_t^{-1}-1)\epsilon_t)\\
&= W_{t-1}\times((\beta_t-1)(1-\epsilon_t)+\beta_t^{-2}\epsilon_t-\beta_t^{-1}\epsilon_t)\\
&\le W_{t-1}\times((\beta_t-1)(1-\epsilon_t)+\beta_t^{-1}\epsilon'_t-\beta_t^{-1}\epsilon_t)\\
&= W_{t-1}\times(\beta^{-1}\epsilon'_t+(1-\epsilon'_t)\beta_t-1+(\epsilon'_t\beta_t-\epsilon_t\beta_t+\epsilon_t-\epsilon_t\beta_t^{-1}))\\
&\le W_{t-1}\times(\beta^{-1}\epsilon'_t+(1-\epsilon'_t)\beta_t-1+(2\epsilon'_t\beta_t-2(\epsilon'_t)^2(\beta_t+\beta_t^{-1})))\\
&= W_{t-1}\times(\beta^{-1}\epsilon'_t+(1-\epsilon'_t)\beta_t-1)\\
&= W_{t-1}\times(2\sqrt{\epsilon'_t(1-\epsilon'_t)}-1) \;=\; W_{t-1}\times(\sqrt{1-(1-2\epsilon'_t)^2}-1) \;\le\; W_{t-1}\times\frac{-(1-2\epsilon'_t)^2}{2}\\
&= W_{t-1}\times\frac{-1-(1-2\gamma_t)+2\sqrt{1-2\gamma_t}}{2} \;\le\; W_{t-1}\times\frac{(-2+2\gamma_t)+(2-2\gamma_t-\gamma_t^2)}{2}\\
&= W_{t-1}\times\left(-\frac{\gamma_t^2}{2}\right).
\end{aligned}
$$

That is, $\widetilde{W}'_t$ (hence $\widetilde{W}_t$) gets decreased from $\widetilde{W}_{t-1}$ at least by $W_{t-1}(\gamma_t^2/2)$.

Now suppose that W_{t-1} is bigger than ϵ. (Otherwise we are done.) Then $\widetilde{W}_t$ decreases from $\widetilde{W}_{t-1}$ by at least $(\epsilon/2)\gamma_t^2$. On the other hand, $\widetilde{W}_0 = W_0 = 1$. Therefore, if $W_t\ge\epsilon$ for all t, $1\le t<T$, then we have $W_T\le\widetilde{W}_T\le 1-(\epsilon/2)\sum_{1\le i\le T}\gamma_i^2$, as claimed. □

Two remarks from the above proof. First consider the reason why the further modification of **MadaBoost** is necessary. This is because we estimated $\Delta(x)$ somewhat larger. More specifically, for the case $x\in Q$, we estimated $\Delta(x)\le\max(\beta_t^{-1}\alpha_t,\beta_t^{-1}-1)w_{t-1}(x)\le\beta_t^{-1}(\beta_t^{-1}-1)w_{t-1}(x)$; but we usually would be able to use the smaller bound $\Delta(x)\le(\beta_t^{-1}-1)w_{t-1}(x)$. Suppose that this bound worked for all (or almost all) $x\in Q$, then we would be able to bound Δ by $W_{t-1}\times(\beta^{-1}\epsilon_t+(1-\epsilon_t)\beta_t-1)=W_{t-1}\times(2\sqrt{\epsilon_t(1-\epsilon_t)}-1)$, which gives $\Delta\le W_{t-1}\times(\sqrt{1-4\gamma_t^2}-1)\le W_{t-1}\times(-2\gamma_t^2)$. Then a similar boosting property would be provable for **MadaBoost**.

Next consider the possibility of proving a faster convergence like **AdaBoost**. The reason of our slower convergence is due to the additive decrement of $\widetilde{W}_t$, whereas the weight W_t gets decreased multiplicatively in **AdaBoost**. Note, however, $\widetilde{W}_t$ gets decreased multiplicatively if we may assume that $\widetilde{W}_t \approx W_t$. In fact, assuming that $\widetilde{W}_t \approx W_t$, we would have Δ ($\stackrel{\text{def}}{=} \widetilde{W}_t - \widetilde{W}_{t-1}$) $\leq \widetilde{W}_{t-1} \times (-\gamma_t^2/2)$. Thus, $\widetilde{W}_t \leq \widetilde{W}_{t-1} \times (1 - \gamma_t^2/2)$; that is, $\widetilde{W}_t$ gets decreased multiplicatively and indeed by a factor similar to **AdaBoost**. Since we may assume $\widetilde{W}_t \approx W_t$ for the first several boosting steps, we can expect convergence speed similar to **AdaBoost**, at least, for the first several steps. This phenomenon has been observed in our experiments [DW99b].

4 Using MadaBoost with Filtering

In the previous section, we described **MadaBoost** and proved that some version of **MadaBoost** is in fact a boosting algorithm. For the comparison with **AdaBoost**, the explanation was given in the subsampling framework. Our original goal, however, was to propose an adaptive boosting technique that can be used under the filtering framework. Here we show that **MadaBoost** indeed satisfies our goal. (Though we have not formally proved the general boosting property for **MadaBoost**, we will use, for the sake of simplicity, **MadaBoost** for our explanation this and the later sections. The same analysis also holds for **MB:1/2**.)

4.1 Our Implementation

Note first that the boosting property of Theorem 2 holds in the filtering framework. The difference is that we consider the error probability on the whole domain X and that the initial distribution D_0 is the original distribution D that we assume over the domain X. More specifically, Lemma 3 holds in the filtering framework, and the error probability of the combined hypothesis (under D over X) is bounded by W_t. The main issue of the filtering framework is the way to generate examples under given distributions and the probability that the filter outputs an example.

Figure 1 gives the description of our example generator or filtering procedure used at the tth boosting round, which is standard and essentially the same as the one given in [Fre95]. Here $w_{t-1}(x)$ and W_{t-1} are those defined in the previous section. But since we are discussing in the filtering framework, $D_0 = D$ (in the definition of $w_{t-1}(x)$), and $W_{t-1} = \sum_{x \in X} w_{t-1}(x)$. The following lemma is immediate from this description.

Lemma 4 *Consider the execution of* $\mathbf{FiltEX}_{D_t}$ *for some* $t \geq 1$.

(1) $\mathbf{FiltEX}_{D_t}$ *outputs* x *with probability* $p(x)D(x)$ $(= w_{t-1}(x))$. *Thus,* $\mathbf{FiltEX}_{D_t}$ *generates examples under probability distribution* $D_t(x) = w_{t-1}(x)/W_{t-1}$.

(2) The probability that one example is output by $\mathbf{FiltEX}_{D_t}$ *is* $\sum_{x \in X} p(x)D(x) = W_{t-1}$.

The above statement (1) guarantees that this filtering procedure $\mathbf{FiltEX}_{D_t}$ gives an example under the desired distribution D_t. On the other hand, it follows from the statement (2) that

Procedure FiltEX$_{D_t}$

begin

if $W_{t-1} < \epsilon$ **then**

output "accurate enough" and exit

end

repeat

use $\mathbf{EX}_{D,f_*}$ to generate an example (x, b);

$p(x) \leftarrow \min\left(\prod_{1\le i\le t-1} \beta_i^{\mathrm{cons}(h_i,x)}, 1\right)$;

% I.e., $p(x) = w_{t-1}(x)/D_0(x)$, where $D_0(x) = D(x)$.

accept (x, b) with probability $p(x)$;

until some example (x, b) is accepted;

return (x, b);

end-procedure.

Figure 1: An example generator **FiltEX**$_{D_t}$ for the filtering framework.

the expected number of executions of $\mathbf{EX}_{D,f_*}$ until some example is accepted (i.e., the expected running time of **FiltEX**$_{D_t}$) is $1/W_{t-1}$. Thus, the procedure takes longer and longer time when W_{t-1} gets decreased. But also recall that W_{t-1} bounds the error probability of the combined hypothesis; hence, we can stop the filtering procedure if it takes, say, more than $1/\epsilon$ steps to get one example. Therefore, we may assume that the running time of the procedure is $\mathcal{O}(1/\epsilon)$. This is the key of our new weighting scheme.

To completely show that **MadaBoost** can be used in the filtering framework, we also need to consider the following three points:- (i) how to set the confidence values so that the overall algorithm does not fail with probability larger than $1-\delta$, (ii) the way to determine "$W_{t-1} < \epsilon$" in **FiltEX**$_{D_t}$, and (iii) the way to estimate γ_t, the advantage of an obtained weak hypothesis h_t.

To solve (i), notice that the places where the algorithm can fail are three:- in one of the executions of **WeakLearn**, in estimating γ_t, and in determining when to stop (i.e., when W_t becomes less than ϵ). It is easy to see that in order to obtain an overall confidence of δ, it suffices to guarantee that any of these two procedures fails with probability at most $\delta_t \stackrel{\mathrm{def}}{=} \delta/(3t(t+1))$ at each boosting step t.

For (ii), we simply count the number of unsuccessful repeat-iterations in the execution of **FiltEX**$_{D_t}$. We determine $W_{t-1} < \epsilon$ if the number exceeds a certain bound that can be easily derived from an appropriate concentration bound. For instance, it follows from the Hoeffding bound that if $W_{t-1} > \epsilon$, then **FiltEX**$_{D_t}$ executes, with probability $\ge 1-\delta_t$, the repeat-iteration more than $\mathcal{O}((1/\epsilon)\ln(1/\delta_t))$ times to yield its output. Thus, when the repeat-iteration is executed $c(1/\epsilon)\ln(1/\delta_t))$ times (where $c > 0$ is some appropriate constant), we can decide $W_{t-1} < \epsilon$ and the probability of making a mistake with this decision is bounded by δ_t.

Finally, we consider (iii), the way to estimate the advantage γ_t of an obtained weak hypothesis h_t. This γ_t is important because it is used to determine the next weight function w_t and to

define the tth combined hypothesis. Recall that in the proof of Theorem 2, we assumed that γ_t was computed exactly. While γ_t can be directly computed under the subsampling framework, this cannot be done in a straightforward manner in the filtering framework. Notice here, however, that if we could obtain an estimate $\widehat{\gamma}_t$ of γ_t such that $|\gamma_t - \widehat{\gamma}_t| \leq \gamma_t/2$, then we could use $\widehat{\gamma}_t/2$ for γ_t. Since this implies that $\gamma_t/4 < \widehat{\gamma}_t/2 < \gamma_t$, the proof of Theorem 2 follows, yielding a slightly worse bound that now depends on $\gamma_t/4$ instead of γ_t. That is, we just slow down a bit the speed of convergence to the desired error. Now how can we obtain such an estimator $\widehat{\gamma}_t$ of γ_t? The situation is different from estimating W_{t-1}, and a straightforward application of a convergence bound like the Chernoff and Hoeffding bounds does not work. Fortunately, we can make use of more sophisticated estimation methods called "adaptive sampling techniques" that have been proposed in [DGW98, DGW99]. (An adaptive sampling technique for estimating the advantage is explicitly explained in a survey paper [Wat00].) By using one of these techniques, it is possible to obtain a desired $\widehat{\gamma}_t$ from $\mathcal{O}((1/\gamma_t^2)\ln(1/\delta_t))$ examples randomly generated by $\mathbf{EX}_{D,f_*}$.

Now summarizing the above discussion, we have the following theorem.

Theorem 5 *Suppose that for given inputs $\epsilon > 0$ and $\delta < 1$, the algorithm* **MadaBoost** *used in the filtering framework as described above executes* **WeakLearn***, finishes T boosting rounds obtaining weak hypotheses with advantages $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_T > 0$, and terminates after the Tth round.*

(1) With probability $1-\delta$, we have (i) $W_{t-1} \geq \epsilon$ for any t, $1 \leq t < T$, and (ii) $W_T < \epsilon$. Thus, $\mathrm{error}_D(X, f_T) \stackrel{\mathrm{def}}{=} \Pr_{x:D}\{\, f_T(x) \neq f_*(x)\,\} \leq W_T < \epsilon$ from Lemma 3. (Below we assume that both (i) and (ii) hold.)

(2) We have

$$\mathrm{error}_D(X, f_T) \;\leq\; W_T \;\leq\; 1 - \sum_{i=1}^{T} \epsilon\gamma_i^2/8.$$

(3) For each execution of $\mathbf{FiltEX}_{D_t}$, $\mathbf{EX}_{D,f_*}$ is called $\mathcal{O}((1/\epsilon)\ln(1/\delta_t))$ times, where $\delta_t \stackrel{\mathrm{def}}{=} \delta/3t(t+1)$. Thus, if **WeakLearn** *requests n_t examples at the tth boosting round, then the number of actual examples generated by $\mathbf{EX}_{D,f_*}$ is $\mathcal{O}((n_t/\epsilon)\ln(1/\delta_t))$. Similarly, for estimating the advantage of the obtained tth weak hypothesis, $\mathbf{EX}_{D,f_*}$ is called $\mathcal{O}((1/\epsilon\gamma_t^2)((\ln(1/\delta_t))^2)$ times.*

4.2 Why Not AdaBoost?

We have just seen that **MadaBoost** works for the filtering framework. At this point, we can also explain clearly why **AdaBoost** cannot be used for the filtering framework.

As it happens with **MadaBoost**, the boosting property of **AdaBoost**, i.e., Theorem 1, still holds in the filtering framework. Furthermore, W_{t-1} also bounds the generalization error of the tth combined hypothesis [2]. On the other hand, we cannot use the above filtering procedure as it

[2]This property only holds under the weighting scheme as defined in Section 2.2.

is because the weights are not bounded between 0 and 1. There are two ways to get around this, either we normalize the weights of **AdaBoost** or we use the original definition in [FS97] where the weights are bounded between 0 and 1. Both ways will lead to the same conclusion; that is, the probability that the filter outputs one example could be exponentially smaller than the current generalization error. We derive this conclusion using the original **AdaBoost** weights and leave the other calculation to the reader. Recall that we are using $\beta_i = \sqrt{\epsilon_i/(1-\epsilon_i)}$. Thus, under this definition, the original weight of AdaBoost as defined in [FS97] is $w_{t-1}^{AB}(x) = \prod_{1\leq i<t} \beta_i^{1+\mathrm{cons}(h_i,x)}$ and $W_{t-1}^{AB} = \sum_{x\in X} w_{t-1}^{AB}(x)$. Moreover, it follows from the definitions that $\mathrm{error}_D(X, f_t) \leq W_{t-1}^{AB}/B_{t-1}$, where $B_{t-1} \stackrel{\mathrm{def}}{=} \prod_{1\leq i<t} \beta_i^2$. Hence, to drive the generalization error down to certain ϵ, we need reduce W_t^{AB} below ϵB_{t-1}. On the other hand, for the obvious filter for **AdaBoost** (the one stated in Figure 1 but using $p(x) = w_{t-1}^{AB}(x)$), Lemma 4 also holds, and hence, it takes $\mathcal{O}(1/W_{t-1}^{AB})$ expected time to output one example. Thus, the expected running time of the filter becomes $\mathcal{O}(1/(\epsilon B_{t-1}))$. To understand the significance of this bound, suppose that the advantage of all obtained weak hypotheses is γ; then $B_{t-1} = ((1-2\gamma)/(1+2\gamma))^{(t-1)}$. Noting that $t = \mathcal{O}((\ln\epsilon)/\gamma^2)$, it is easy to see that the running time of the filter becomes exponential in $1/\gamma$. This is why we needed to modify the weighting scheme of **AdaBoost**.

5 Noise Tolerance

We have argued that one of the drawbacks of **AdaBoost** is that it does not seem to be noise resistant. While from the theoretical side nothing is known about it, recent work of Dietterich [Die98] provides a reasonable experimental evidence that **AdaBoost** is not noise resistant. A very plausible explanation for this phenomenon, also verified experimentally by Dietterich in his study, is that **AdaBoost** seems to place more and more weight on the noisy examples making some of the weak classifiers to agree on them and degradating the overall performance of the combined hypothesis. Notice that with our new weighting scheme, the weights never become bigger than its original value. Thus, with our algorithm we may be able to avoid the problem detected in **AdaBoost**. We are planning to study experimentally this point in future work.

For the time being, we show here, from the theoretical point of view, that **MadaBoost** is in fact resistant to certain kinds of noise by showing that it belongs to the statistical query model of learning introduced by Kearns [Kea93]. Whether **AdaBoost** or any of its variants fall into this model or not is not known. Notice that one of the key points for showing this result is the fact that **MadaBoost** works in the filtering framework and thus, we can in fact estimate probabilities from the modified distributions.

Before showing the result, let us recall the definitions of the statistical query model. In the *statistical query model* (in short, SQ model) the learner does not have access to examples any more. The example oracle $EX(f_*, D)$ is replaced by a statistics oracle $STAT(f_*, D)$ that answers to statistical queries. A *statistical query* is of the form $[\chi, \tau]$, where τ is a tolerance parameter, and χ is a mapping from labeled examples to $\{0,1\}$ (i.e. $\chi : X \times \{0,1\} \to \{0,1\}$) corresponding to an indicator function for those examples about which statistic are to be collected. To the

query $[\chi, \tau]$, the statistics oracle returns $STAT(f_*, D)[\chi, \tau]$, that is an estimate $\widehat{P}_\chi$ of $P_\chi = \Pr_{x:D}\{\, \chi(x, f_*(x)) \,\}$ that satisfies $|\widehat{P}_\chi - P_\chi| \leq \tau$. The intuition behind this query type is the following. In the usual PAC learning model, a learner has to make a decision based on observed examples. Thus, if the learner is given noisy examples, its decision based on them could be totally wrong. On the other hand, if a learner uses information on statistics about the target concept that are taken from a sufficiently big amount of examples, then the learner can still obtain reasonable estimates even from noisy examples and thus, it can make a correct decision based on them. By the SQ model we can discuss this situation. In fact, this intuition was shown to be correct by Kearns [Kea93]. He showed that, if the noise rate is at most η, then every statistical query could be simulated using $\mathcal{O}(1/(\tau(1-2\eta)))$ noisy examples (ignoring dependencies on the accuracy and confidence parameters) and thus, any learning algorithm using only statistical queries is resistant to random classification noise. Moreover, nearly every known PAC learning algorithm can be shown to belong to the SQ model, showing the generality of the framework. Further work by Decatur [Dec93] has also shown that an statistical query algorithm is resistant to several other kinds of noise.

Now we show that **MadaBoost** belongs to the SQ model. For this, let us assume that **WeakLearn** is a SQ learning algorithm; that is, it uses a statistical oracle instead of an example oracle. Notice here that **WeakLearn** works under the modified distributions. That is, at each step t, it does not use $STAT(f_*, D)$ but $STAT(f_*, D_t)$. On the other hand, the boosting algorithm is given only the statistical oracle for the original distribution D. Therefore our task here is to show the way to simulate a query to $STAT(f_*, D_t)$ by using queries to $STAT(f_*, D)$. First we show how we can rewrite a statistical query to $STAT(f_*, D_t)$ in terms of queries to $STAT(f_*, D)$. Next we show how precise the queries to $STAT(f_*, D)$ could be made so the tolerance required by $STAT(f_*, D_t)$ is met.

To be specific, for the rest of the discussion, let us assume that we are in the step t of the boosting process and that **WeakLearn** is making a query $STAT(f_*, D_t)[\chi, \tau]$. Then we discuss the way to simulate this query $STAT(f_*, D_t)[\chi, \tau]$. Also we assume here that $W_t \geq \epsilon$ for a given parameter ϵ to the boosting algorithm, because otherwise, the boosting process would have been terminated.

We need to divide a subspace $X' \stackrel{\text{def}}{=} \{\, x \in X \mid f_t(x) = f_*(x) \,\}$ in slices depending on the value of $w_t(x)$. For this, we define n as the smallest integer satisfying the following inequality.

$$\log\left(\frac{96(n+1)^3}{\tau\epsilon}\right) \;\leq\; n.$$

Let l be the smallest integer such that $Y^l \leq \prod_{i=1}^{t} \beta_i$ holds, where Y is defined as

$$Y \;\stackrel{\text{def}}{=}\; 1 - \frac{\tau\epsilon}{192(n+1)^2}.$$

Then define the following set for each $k \geq 0$.

$$E_k \stackrel{\text{def}}{=} \{\ x \mid x \in X \text{ and } Y^{k+1} \leq \prod_{i=1}^{t} \beta_i^{\text{cons}(h_i,x)} < Y^k\ \}.$$

Note that $\{E_k\}_{k\geq 0}$ is the division of X'.

Notice that the above definition makes sense since n always exists and Y is strictly smaller than 1. Armed with this definition, we can now rewrite the query $STAT(f_*, D_t)[\chi, \tau]$ as follows.

$$\begin{aligned}
STAT(f_*, D_t)[\chi,\tau] &= \Pr_{x:D_t}\{\ \chi(x, f_*(x))\ \} \\
&= \sum_{x\in X} \chi(x, f_*(x)) D_t(x) \\
&= \sum_{\substack{x\in X \\ f_t(x)=f_*(x)}} \chi(x, f_*(x)) D_t(x) + \sum_{\substack{x\in X \\ f_t(x)\neq f_*(x)}} \chi(x, f_*(x)) D_t(x) \\
&= \sum_{0\leq k\leq l-1} \sum_{x\in E_k} \chi(x, f_*(x)) D_t(x) + \sum_{\substack{x\in X \\ f_t(x)\neq f_*(x)}} \chi(x, f_*(x)) \frac{D(x)}{W_t} \\
&= \sum_{0\leq k\leq l} \Pr_{x:D_t}\{\ \chi(x, f_*(x)) \wedge x \in E_k\ \} + \frac{\Pr_{x:D}\{\ \chi(x, f_*(x)) \wedge f_t(x) \neq f_*(x)\ \}}{W_t}
\end{aligned}$$

Now we want to approximate these two additive terms of the above formula. We show how each term can be approximated efficiently up to an additive tolerance of $\tau/2$; then the overall probability is approximated up to an additive tolerance of τ as desired.

The second term in the above is already given in terms of probability with respect to D; hence, we can use $STAT(f_*, D)$ to approximate it appropriately as shown by the following lemma. (The proof is easy and omitted here.)

Lemma 6 *Let $p = \Pr_{x:D}\{\ \chi(x, f_*(x)) \wedge f_t(x) \neq f_*(x)\ \}$ and let $\widehat{p}$ and $\widehat{W_t}$ be two estimators such that*

$$|\,\widehat{p} - p\,| \ \leq\ \tau W_t/6 \quad \textit{and} \quad |\,\widehat{W_t} - W_t\,| \ \leq\ \tau W_t/6,$$

then an estimator $\widehat{p}/\widehat{W_t}$ satisfies

$$\left| \frac{\widehat{p}}{\widehat{W_t}} - \frac{p}{W_t} \right| \ \leq\ \frac{\tau}{2}.$$

For the first term, let us define the following probabilities in order to simplify the notation.

$$\begin{aligned}
p_k^{(1)} &\stackrel{\text{def}}{=} \Pr_{x:D_t}\{\ \chi(x, f_*(x)) \wedge x \in E_k\ \}, \quad \text{and} \\
p_k^{(2)} &\stackrel{\text{def}}{=} \Pr_{x:D}\{\ \chi(x, f_*(x)) \wedge x \in E_k\ \}
\end{aligned}$$

Then for any $k \geq 0$, the following inequalities hold by definition of E_k.

$$\frac{Y^{k+1} p_k^{(2)}}{W_t} \ \leq\ p_k^{(1)} \ \leq\ \frac{Y^k p_k^{(2)}}{W_t}. \tag{1}$$

Thus, our goal is to obtain, for every k, an estimator $\widehat{p}_k^{(1)}$ that approximates $p_k^{(1)}$ up to an appropriate additive tolerance. We estimate $p_k^{(1)}$ by using the estimators $\widehat{p}_k^{(2)}$ and $\widehat{W_t}$ of $p_k^{(2)}$ and W_t. The following lemma states the way.

Lemma 7 *For any $k \geq 0$, let $\widehat{p}_k^{(2)}$ and $\widehat{W}_t$ two estimators satisfying*

$$|\,\widehat{p}_k^{(2)} - p_k^{(2)}\,| \;\leq\; \frac{\tau W_t}{24Y^{k+1}(n+1)} \quad \textit{and} \quad |\,\widehat{W}_t - W_t\,| \;\leq\; \frac{\tau W_t}{24(n+1)}.$$

Then by defining $\widehat{p}_k^{(1)} \stackrel{\text{def}}{=} Y^{k+1}\widehat{p}_k^{(2)}/\widehat{W}_t$, we have

$$|\,\widehat{p}_k^{(1)} - p_k^{(1)}\,| \;\leq\; \frac{\tau}{4(n+1)}.$$

Proof. Since $W_t \geq \epsilon$, we have $1 - Y = \tau\epsilon/(192(n+1)^2) \leq \tau W_t/8(n+1)$. Moreover, since $p_k^{(2)}$ and Y^k are both less than or equal to 1, the following inequality holds.

$$Y^k(1-Y)p_k^{(2)} \;\leq\; \frac{\tau W_t}{8(n+1)},$$

which implies that

$$\frac{Y^k p_k^{(2)}}{W_t} - \frac{Y^{k+1}p_k^{(2)}}{W_t} \;\leq\; \frac{\tau}{8(n+1)}.$$

This inequality together with inequality (1) implies

$$p_k^{(1)} - \frac{\tau}{4(n+1)} \;\leq\; \frac{Y^{k+1}p_k^{(2)}}{W_t} - \frac{\tau}{8(n+1)} \quad \text{and} \quad \frac{Y^{k+1}p_k^{(2)}}{W_t} + \frac{\tau}{8(n+1)} \;\leq\; p_k^{(1)} + \frac{\tau}{4(n+1)}.$$

Thus, if we show that estimator $\widehat{p}_k^{(1)}$ as defined in the statement of the lemma approximates $Y^{k+1}p_k^{(2)}/W_t$ up to an additive tolerance smaller than $\tau/(8(n+1))$, then, by the inequality above, we can conclude that $\widehat{p}_k^{(1)}$ also approximates $p_k^{(1)}$ up to an additive tolerance smaller than $\tau/(4(n+1))$ as claimed in the statement of the lemma.

First note the following two inequalities, which are easy to show.

$$\frac{Y^{k+1}p_k^{(2)} + \frac{\tau W_t}{24(n+1)}}{W_t - \frac{\tau W_t}{24(n+1)}} \;\leq\; \frac{Y^{k+1}p_k^{(2)}}{W_t} + \frac{\tau}{8(n+1)} \quad \text{and} \quad \frac{Y^{k+1}p_k^{(2)}}{W_t} - \frac{\tau}{8(n+1)} \;\leq\; \frac{Y^{k+1}p_k^{(2)} - \frac{\tau W_t}{24(n+1)}}{W_t + \frac{\tau W_t}{24(n+1)}}.$$

Next use the assumption of the lemma on $\widehat{p}_k^{(2)}$ and $\widehat{W}_t$; then we have

$$\frac{Y^{k+1}p_k^{(2)} - \frac{\tau W_t}{24(n+1)}}{W_t + \frac{\tau W_t}{24(n+1)}} \;\leq\; \frac{Y^{k+1}\widehat{p}_k^{(2)}}{\widehat{W}_t} \;\leq\; \frac{Y^{k+1}p_k^{(2)} + \frac{\tau W_t}{24(n+1)}}{W_t - \frac{\tau W_t}{24(n+1)}}.$$

This implies that $|\widehat{p}_k^{(1)} - Y^{k+1}p_k^{(2)}/W_t| \leq \tau/8(n+1)$, proving the lemma as already argued. □

Notice that as k becomes large the probability $p_k^{(1)}$ gets smaller. In fact, we do not need to approximate all of them; only the first $n+1$ would be enough. The rest can be discarded as shown in the following lemma.

Lemma 8 $\sum_{n+1 \leq k \leq l-1} p_k^{(1)} \;\leq\; \tau/4.$

Proof. Since $Y < 1$ and $\epsilon \leq W_t$, it follows from our choice of n that $\log(96(k+1)^3/(\tau W_t)) \leq k \log(1/Y)$ for any $k > n$. This implies

$$\frac{Y^k}{W_t} \;\leq\; \frac{\tau}{96(k+1)^3} \;\leq\; \frac{\tau}{4k^2}.$$

Moreover, the following chain of inequalities also hold.

$$\sum_{n+1\leq k\leq l-1} p_k^{(1)} \;\leq\; \sum_{n+1\leq k\leq l-1} \frac{Y^k p_k^{(2)}}{W_t} \;\leq\; \sum_{n+1\leq k\leq l-1} \frac{Y^k}{W_t} \;\leq\; \sum_{n+1\leq k\leq l-1} \frac{\tau}{4k^2}.$$

Then the lemma follows by noticing that $\sum_{n+1\leq k\leq l-1} \tau/4k^2$ is smaller than $\tau/4$. $\square$

Thus, for $1 \leq k \leq n$, we use the estimator $\widehat{p}_k^{(1)}$ as defined in Lemma 7, and for any $k > n$, we just use $\widehat{p}_k^{(1)} = 0$ as our estimator. In this way, we can bound the error as follows.

$$\begin{aligned}\left| \sum_{0\leq k\leq l-1} p_k^{(1)} - \sum_{0\leq k\leq l-1} \widehat{p}_k^{(1)} \right| \;&\leq\; \sum_{0\leq k\leq n} |p_k^{(1)} - \widehat{p}_k^{(1)}| + \sum_{n+1\leq k\leq l-1} |p_k^{(1)} - 0| \\ &\leq\; \sum_{0\leq k\leq n} \frac{\tau}{4(n+1)} + \sum_{n+1\leq k\leq l-1} \frac{\tau}{4k^2} \;\leq\; \frac{\tau}{2}.\end{aligned}$$

Estimators $\widehat{p}$ and $\widehat{p}_k^{(2)}$ can be obtained from queries to $STAT(f_*, D)$, since they are estimators of probabilities with respect to D, and the conditions $f_t(x) \neq f_*(x)$ and $x \in E_k$ can be tested in polynomial time. On the other hand, $\widehat{W}_t$ cannot be directly obtained from $STAT$. Before describing how to obtain $\widehat{W}_t$, let us discuss about the tolerances of these estimators.

We need to verify that the tolerances required for all the approximations involved in the proof are not too small. In other words, we need to show that the inverse of all the tolerances are polynomial in $1/\epsilon$ and $1/\tau$. First recall that all the tolerances are required to depend on W_t. This value is not known but we can assume that it is larger than ϵ throughout all the boosting process; hence, we can substitute W_t by ϵ in all the tolerances. The tolerance required for $\widehat{W}_t$ in Lemma 7 (i.e., $\tau\epsilon/24(n+1)$) is the smallest among all the tolerances required; thus, it is enough to show that $24(n+1)/\tau\epsilon$ is bounded by a polynomial in $1/\epsilon$ and $1/\tau$. But this follows from the fact that n is bounded by a polynomial; more specifically, it is easy to show that n is bounded by $\mathcal{O}((1/\epsilon\tau)\ln(1/\epsilon\tau))$.

It remains to see how can we approximate W_t from $STAT(f_*, D)$. The smallest tolerance required for W_t through the proof is $\tau\epsilon/24(n+1)$; hence, we show how to approximate W_t up to this tolerance. First we rewrite W_t as follows.

$$W_t \;=\; \sum_{x\in X} w_t(x) \;=\; \Pr_{x:D}\{\, f_t(x) \neq f_*(x) \,\} + \sum_{0\leq k\leq l-1} \;\sum_{x\in E_k} w_t(x).$$

Moreover, by definition of E_k, the following inequality holds for any $k \geq 0$.

$$Y^{k+1} q_k^{(2)} \;\leq\; q_k^{(1)} \;\leq\; Y^k q_k^{(2)},$$

where $q_k^{(2)} = \Pr_{x:D}\{\, x \in E_k \,\}$ and $q_k^{(1)} = \sum_{x\in E_k} w_t(x)$.

The following lemma shows how to approximate W_t.

Lemma 9 *Let* $q = \Pr_{x:D}\{\, f_t(x) \neq f_*(x)\,\}$ *and for any* $k \geq 0$, *let* $\widehat{q}_k^{(2)}$ *and* $\widehat{q}$ *be two estimators such that*

$$|\,\widehat{q}_k^{(2)} - q_k^{(2)}\,| \;\leq\; \frac{\tau\epsilon}{Y^{k+1}192(n+1)^2} \quad \textit{and} \quad |\,\widehat{q} - q\,| \;\leq\; \frac{\tau\epsilon}{48n}.$$

Then by defining $\widehat{W}_t \stackrel{\text{def}}{=} \widehat{q} + \sum_{0\leq k\leq n} \widehat{q}_k^{(2)}$, *we have*

$$|\,W_t - \widehat{W}_t\,| \;\leq\; \frac{\tau\epsilon}{24(n+1)}.$$

Proof. The proof is similar to the previous two proofs; thus, we only give its sketch. First by our choice of Y, the following holds.

$$Y^k q_k^{(2)} - Y^{k+1} q_k^{(2)} \;\leq\; \frac{\tau\epsilon}{192(n+1)^2},$$

which implies

$$q_k^{(1)} - \frac{\tau\epsilon}{96(n+1)^2} \;\leq\; Y^{k+1}q_k^{(2)} - \frac{\tau\epsilon}{192(n+1)^2} \quad \text{and} \quad Y^{k+1}q_k^{(2)} + \frac{\tau\epsilon}{192(n+1)^2} \;\leq\; q_k^{(1)} + \frac{\tau\epsilon}{96(n+1)^2}.$$

Thus, from the assumption on the estimator $\widehat{q}_k^{(2)}$, it follows that $Y^{k+1}\widehat{q}_k^{(2)}$ approximates $q_k^{(1)}$ up to an additive tolerance of $\tau\epsilon/96(n+1)^2$. Moreover, by our choice of n, we can show that the value of $q_k^{(1)}$ for any $k > n$ is smaller than $\epsilon\tau/96(k+1)^3$. Then we now obtain the following bound.

$$\begin{aligned}
|\,W_t - \widehat{W}_t\,| &= \left|\left(q + \sum_{0\leq k\leq l-1} q_k^{(1)}\right) - \left(\widehat{q} + \sum_{0\leq k\leq n} \widehat{q}_k^{(2)}\right)\right| \\
&\leq |\,q - \widehat{q}\,| + \sum_{0\leq k\leq n} \left|\, q_k^{(1)} - \widehat{q}_k^{(2)} \,\right| + \sum_{n+1\leq k\leq l-1} p_k^{(1)} \\
&\leq \frac{\tau\epsilon}{48(n+1)} + \sum_{0\leq k\leq n} \frac{\tau\epsilon}{96(n+1)^2} + \sum_{n+1\leq k\leq l-1} \frac{\tau\epsilon}{96(k+1)^2(n+1)} \;\leq\; \frac{\tau\epsilon}{24(n+1)}.
\end{aligned}$$

□

Summarizing, we have shown how to estimate $\Pr_{x:D_t}\{\, \chi(x, f_*(x))\,\}$ up to tolerance τ by combining appropriately the answers of the following statistical queries.

$STAT(f_*, D)[\, \chi(x, f_*(x)) \wedge f_t(x) \neq f_*(x)\,,\, \tau\epsilon/6\,]$
$STAT(f_*, D)[\, \chi(x, f_*(x)) \wedge x \in E_k\,,\, \tau\epsilon/(24Y^{k+1}(n+1))\,]$ (for each k, $0 \leq k \leq n$)
$STAT(f_*, D)[\, f_t(x) \neq f_*(x)\,,\, \tau\epsilon/(48n)\,]$, and
$STAT(f_*, D)[\, x \in E_k\,,\, \tau\epsilon/(192Y^{k+1}(n+1)^2)\,]$ (for each k, $0 \leq k \leq n$).

Therefore, we have proven the following theorem. (The same statement holds for **MB:1/2**.)

Theorem 10 *Algorithm* **MadaBoost** *is an statistical query boosting algorithm.*

6 Concluding Remarks

We have presented **MadaBoost**, a modification of **AdaBoost** that enjoys some properties that **AdaBoost** lacked of, namely its ability to run under the filtering framework and to belong to the statistical query learning model. Our modification for **MadaBoost** is very simple, we just keep the weights of the examples bounded by its initial value and the rest of the algorithm is the same as **AdaBoost**. We also considered a further modified version **MB:1/2** of **MadaBoost** in which the algorithm define the parameter β_t by using the half of the real advantage of the obtained weak hypothesis h_t. We have shown that **MB:1/2** is a boosting algorithm in the PAC sense assuming that the advantage sequence is monotonically decreasing. We believe that **MadaBoost** has a similar property; furthermore, while the convergence speed we proved is exponential slower than that of **AdaBoost**, we believe that **MadaBoost** (and **MB:1/2**) has a similar convergence speed at least for a certain number of initial boosting steps. (Our experiments support these conjectures.) Certainly, proving these two conjectures is intriguing and challenging from the theoretical view point.

We think that the ability of **MadaBoost** to run efficiently under the filtering framework makes a large difference compared to **AdaBoost** when a large amount of data is available. To confirm this, we have performed an experimental comparison of **MadaBoost** (and **MB:1/2**) and **AdaBoost** in a companion paper [DW99b] and we briefly summarize those results here.

Our experimental results indicate that, when run under the subsampling framework, these algorithms perform similarly in terms of training and test set error. On training sets, **AdaBoost** converges a bit faster than **MadaBoost**, but then **AdaBoost** generates more difficult distributions, which results in slowing down the convergence speed of **AdaBoost**. In fact, when using a strong base learner like Naive Bayes, the boosting process has to stop earlier when using **AdaBoost** because there are no weak hypotheses available. This results on a slight advantage for **MadaBoost** that can run longer and further reduce the *generalization* error. In all our experiments, the advantage sequence obtained using **MadaBoost** is always monotonically decreasing. That is, it satisfies the assumption of Theorem 2.

When using both algorithms for the filtering framework the differences show up. **AdaBoost** becomes extremely slow in generating a new sample at each iteration while **MadaBoost** not, as predicted by the theoretical results in Section 4. Moreover, for some datasets, this also degradates the behavior of **AdaBoost** compared to the results obtained when running it under the subsampling framework (this has been already observed by Quinlan [Qui96]) while **MadaBoost** performs equally well under both frameworks. We believe that the ability of **MadaBoost** to work under the filtering framework would make some difference in practical applications where we have a huge dataset and we want to speed up its execution through sampling. We have already obtained some results in this direction [DW99a].

The experimental results using noisy data show that there is not so much difference between **AdaBoost** and **MadaBoost** even when we run them under the filtering framework (except for the differences that already show up when using noise free data). However, we have observed

that, even when there is noise, boosting still generally improves the accuracy as opposed to the results presented in [Die98]. The reason for this discrepancy could be that in our experiments we used a different base learner (decision stumps and Naive Bayes) than the one used by Dietterich (C4.5).

References

[AL88] D. Angluin and P. Laird, Learning from noisy examples, *Machine Learning*, 2(4):343–370, 1988.

[AD93] J.A. Aslam and S.E. Decatur, General bounds on statistical query learning and PAC learning with noise via hypothesis boosting, in *Proc. of the 34th Annual Sympos. on Foundations of Comp. Sci.*, 282–291, 1993.

[Bre98] L. Breiman, Arcing Classifiers, *Annals of Statistics*, 26(3):801–849,1998.

[Bre99] L. Breiman, Pasting small votes for classification in large databases and on-line, *Machine Learning*, 36, 85–103, 1999.

[Dec93] S.E. Decatur, Statistical queries and faulty PAC oracles, in *Proc. of the 6th Annual ACM Workshop on Computational Learning Theory*, 1993.

[Die98] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization, *Machine Learning*, 32:1–22, 1998.

[DKM96] T. Dietterich, M. Kearns, and Y. Mansour, Applying the weak learning framework to understand and improve C4.5, in *Proc. 13th International Conference on Machine Learning*, 96–104, 1996.

[DGW98] C. Domingo, R. Gavaldà, and O. Watanabe, Practical algorithms for on-line selection, in *Proc. of the First International Conference on Discovery Science*, DS'98, Lecture Notes in Artificial Intelligence 1532, 150–161, 1998.

[DGW99] C. Domingo, R. Gavaldà, and O. Watanabe, Adaptive sampling methods for scaling up knowledge discovery algorithms, in *Proc. of the Second International Conference on Discovery Science*, DS'99, Lecture Notes in Artificial Intelligence, 1999, to appear.

[DW99a] C. Domingo and O. Watanabe, Scaling up a boosting-based learner via adaptive sampling, PAKDD2000, to appear.

[DW99b] C. Domingo and O. Watanabe, Experimental evaluation of a modified AdaBoost for the filtering framework. Technical report C-139, Dept. of Mathematics and Computer Science, Tokyo Institute of Technology, available from `www.is.titech.ac.jp/research/research-report/C/index.html`, 1999.

[DC96] H. Drucker and C. Cortes, Boosting decision trees, in *Advances in Neural Information Processing Systems 8*, 479–485, 1996.

[Fre95] Y. Freund, Boosting a weak learning algorithm by majority, *Information and Computation*, 121(2):256–285, 1995.

[Fre99] Y. Freund, An adaptive version of the boost by majority algorithm, in *Proc. of the Twelfth Annual Conference on Computational Learning Theory*, 1999.

[FS97] Y. Freund and R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[FS96] Y. Freund and R.E. Schapire, Experiments with a new boosting algorithm, in *Proc. of the Thirteenh International Conference on Machine Learning*, 148–156, 1996.

[FHT98] J. Friedman, T. Hastie, and R. Tibshirani, Additive logistic regression: a statistical view of boosting, Technical report, 1998.

[Kea93] M. Kearns, Efficient noise-tolerant learning from statistical queries, In *Proc. of the Twenty-Fifth Annual ACM Sympos. on Theory of Comput.*, 392–401, 1993.

[KV94] M.J. Kearns and U.V. Vazirani, *An Introduction to Computational Learning Theory*, Cambridge University Press, 1994.

[Qui96] J.R. Quinlan, Bagging, boosting, and C4.5, in *Proc. of the 13th National Conference on Artificial Intelligence*, 725–730, 1996.

[Sch90] R.E. Schapire, The strength of weak learnability, *Machine Learning*, 5(2):197–227, 1990.

[SFBL98] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee, Boosting the margin: A new explanation for the effectiveness of voting methods, *The Annals of Statistics*, 26(5):1651–1686, 1998.

[Wat99] O. Watanabe, From computational learning theory to discovery science, in *Proc. 26th International Colloquium on Automata, Languages and Programming*, ICALP'99, Lecture Notes in Computer Science 1644, 134–148, 1999.

[Wat00] O. Watanabe, Simple sampling techniques for discovery science, IEICE Trans., 2000, to appear.