

title: Algorithmic Aspects of Boosting

author: 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), 1998–2000.

Abstract. We discuss algorithmic aspects of boosting techniques, such as Majority Vote Boosting [Fre95], AdaBoost [FS97], and MadaBoost [DW00a]. Considering a situation where we are given a huge amount of examples and asked to find some rule for explaining these example data, we show some reasonable algorithmic approaches for dealing with such a huge dataset by boosting techniques. Through this example, we explain how to use and how to implement “adaptivity” for scaling-up existing algorithms.

1 Introduction

During the *Discovery Science Project*, the author and his colleagues, Carlos Domingo and Ricard Gavaldà, have developed simple and easy-to-use sampling techniques that would help us scaling up algorithms designed for machine learning and data mining. (For studying our sampling techniques, see, for example, technical papers [DGW01]. Also see survey papers [Wat00a, Wat00b] for adaptive sampling techniques in general.)

A common feature of these sampling techniques is “adaptivity”. Tuning up algorithms is important for making them applicable to heavy computational tasks. It often needs, however, experts’ knowledge on algorithms and tasks, and it is sometimes very difficult even to experts. A typical example is a case where the performance of an algorithm A depends on some parameters and we need to set these parameters appropriately. An expert may be able to set the parameters appropriately (using some complicated formulas), but even an expert may not be able to do so because the choice depends on some *hidden* parameter x that is specific to the task that A is supposed to solve. But as it sometimes occurs in data mining applications, such a hidden parameter x can be detected on the course of solving the task. Then one might hope for an algorithm that infers x and set up the parameters required for A appropriately; that is, an algorithm that *adapts* it to the current circumstances and achieves nearly best performance. We have developed sampling techniques with such adaptivity.

Here we explain how to use and how to implement this “adaptivity” for scaling-up existing algorithms, by using boosting algorithms as example. (In fact, being inspired by AdaBoost [FS97], a boosting technique with some adaptive property, we have studied sampling techniques

for developing a boosting based fully adaptive classifier.) We discuss algorithmic aspects of boosting techniques; we propose some ways to design scalable algorithms based on boosting techniques. Through this explanation, we show simple but useful techniques for adaptive algorithms.

Boosting, or more precisely, a boosting technique, is a way to design a strong PAC learning algorithm by using an assumed weak PAC learning algorithm. The first boosting technique has been introduced by Schapire [Sch90], for showing the equivalence between the strong and weak PAC-learnability notions. Since then, in particular, due to the success of AdaBoost, several boosting techniques have been proposed; see, e.g., the Proceedings of *the Thirteenth Annual Conference on Computational Learning Theory* (2000). Various important investigations have been also made concerning boosting properties; see, e.g., [FHT98, CSS00]. While these statistical investigations are essential for boosting techniques, we could also consider some algorithmic aspects of boosting techniques, such as actual methods to design *efficient* learning algorithms based on boosting techniques, which has not been addressed so much. Here we focus on three boosting techniques — Majority Vote Boosting (which we abbreviate MajBoost) [Fre95], AdaBoost [FS97], and MadaBoost [DW00a] — and discuss how to apply these techniques to design classification algorithms that could be applicable for huge datasets.

2 Boosting Techniques and Our Goal

In this paper, we consider some simple “data mining task” for our example and explain how to design reasonable algorithms based on boosting techniques. Here we specify our example problem; we also review basics of boosting techniques.

Our example problem is simply to find a binary classification rule explaining a given dataset, i.e., a set X of examples. More specifically, X is a set of labeled examples, i.e., a pair $(x, l_*(x))$, where x is an actual attribute data of some *instance* and $l_*(x)$ is the *label* of the instance. The form of data x is not important for our discussion; we may intuitively regard it as a vector consisting of values of a certain set of attributes. On the other hand, it is important for us that $l_*(x)$ takes only two values, either $+1$ or -1 . That is, we consider the binary classification problem.

An important point (we assume here) is that X is huge, and it is not practical even just going through the whole examples in X . We also assume that X reflects the real world quite well without any serious errors; thus, our task is simply to find a good classification rule for X under the uniform distribution on X .

In summary, our task is to find, with the confidence $> 1 - \delta_0$, some binary classification rule for a huge set X of labeled examples whose error on X under the uniform distribution is bounded by ε_0 . Let us call it “our learning goal”. In order to simplify our discussion, we will assume that δ_0 is some constant and ignore the factor for the confidence (which is $\log(1/\delta_0)$) in the following analysis.

Next we review boosting techniques. A boosting technique is a way to design a “strong”

Procedure Generic Boost;

Given: A set S of labeled examples and a distribution D on S ;

begin

$t \leftarrow 1$; $D_1 \leftarrow D$;

repeat

train **WeakLearn** on S under D_t

and obtain a weak hypothesis h_t with advantage γ_t ;

compute a weight α_t of h_t ;

modify D_t and obtain a new distribution D_{t+1} ;

$t \leftarrow t + 1$;

until $t > T$, or the accuracy of a combined hypothesis f_t (see below)

from weak hypotheses h_1, \dots, h_t obtained so far reaches to a certain level;

output the following hypothesis f_T

$$f_T(x) = \text{sign} \left(\sum_{1 \leq i \leq T} \alpha_i h_i(x) \right),$$

where T is the total number of obtained weak hypotheses;

end-procedure.

Figure 1: A Genetic Boosting Algorithm

learning algorithm by using an assumed “weak” learning algorithm **WeakLearn**. (Throughout this paper, we use **WeakLearn** to denote a weak learning algorithm.) Precisely, **WeakLearn** is an algorithm that asks for a certain amount of examples under a fixed but unknown distribution and outputs a weak hypothesis with error probability P_{err} strictly smaller than $1/2$ under the distribution from which the examples were generated. The quantity $P_{\text{err}} - 1/2$ is called the *advantage* of the obtained weak hypothesis.

Almost all boosting techniques follow some common outline. In particular, three boosting techniques we will consider here share the same outline that is stated as a procedure of Figure 1. This boosting procedure runs **WeakLearn** several times, say T times, under distributions D_1, \dots, D_T that are slightly modified from the given distribution D and collects weak hypotheses h_1, \dots, h_T . 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, \dots, h_t have been obtained, the boosting algorithm computes a new distribution D_{t+1} that puts more weight on those instances that have been misclassified by most of h_1, \dots, h_t . Then a new hypothesis h_{t+1} produced by **WeakLearn** on this distribution D_{t+1} should be strong on those problematic instances, thereby improving the performance of the combined hypothesis built from h_1, \dots, h_{t+1} . (In the following, we call each execution of **WeakLearn** a *boosting round* or *boosting step*.)

Boosting techniques differ typically on (i) a way weak hypotheses are combined, and (ii) a weighting scheme used to compute modified distributions. Here we specify these points to define

three boosting techniques MajBoost, AdaBoost, and MadaBoost.

The first point (i) is specified by simply showing how to define the weight α_t used in the above generic procedure.

$$\begin{array}{ll} \text{MajBoost:} & \alpha_t = 1. \\ \text{AdaBoost:} & \alpha_t = \log \beta_t^{-1}. \\ \text{MadaBoost:} & \alpha_t = \log \beta_t^{-1}. \end{array} \quad \left(\text{Where } \beta_t = \sqrt{\frac{1-2\gamma_t}{1+2\gamma_t}} \right)$$

For the second point (ii), we define a weight $w_{t-1}(x)$ of each instance x in S . Then $D_t(x)$ is defined by $D_t(x) = w_{t-1}(x)/W_{t-1}$, where $W_{t-1} = \sum_{x \in S} w_{t-1}(x)$. (Below β_i is defined as above, and $\text{cons}(h_i, x)$ is 1 if $h_i(x) = l_*(x)$, and -1 otherwise.)

MajBoost:

$$w_{t-1}(x) = D(x) \times \binom{T-t}{T/2-r} \left(\frac{1}{2+\gamma} \right)^{T/2-r} \left(\frac{1}{2-\gamma} \right)^{T/2-t+r},$$

where γ is a lower bound of $\gamma_1, \dots, \gamma_T$, and

r is the number of hypotheses h_1, \dots, h_{t-1} misclassifying x .

AdaBoost:

$$w_{t-1}(x) = D(x) \times \prod_{1 \leq i \leq t-1} \beta_i^{\text{cons}(h_i, x)}.$$

MadaBoost:

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

Notice that MajBoost uses the total number T of boosting rounds and a lower bound of advantages γ to compute weights; thus, these values should be determined in advance. (Since there is a formula (for MajBoost) to compute T from γ , what we need to know in advance is only γ .)

Now by applying α_t and D_t defined above in the procedure of Figure 1, we can define MajBoost, AdaBoost, and MadaBoost. The following theorem states the efficiency of these techniques.

Theorem 1 *In the execution of each boosting technique, suppose that the advantage of obtained weak hypotheses are $\gamma_1, \gamma_2, \dots$ and that γ_{lb} is their lower bound. Then for any ε , if T , the total number of boosting rounds, satisfies the following inequality, then the error probability of combined hypothesis f_T on S under the distribution D is less than ε .*

$$\begin{array}{ll} \text{MajBoost:} & T \geq \frac{1}{2\gamma_{lb}^2} \ln \frac{1}{2\varepsilon} \quad (\text{if } \gamma_{lb} \text{ is used for } \gamma), \\ \text{AdaBoost:} & \sum_{1 \leq i \leq T} \gamma_i^2 \geq \frac{1}{2} \ln \frac{1}{\varepsilon} \quad \left(\text{or roughly, } T \geq \frac{1}{2\gamma_{lb}^2} \ln \frac{1}{\varepsilon} \right) \\ \text{MadaBoost:} & \sum_{1 \leq i \leq T} \gamma_i^2 \geq \frac{1}{2} \cdot \frac{1}{\varepsilon} \quad \left(\text{or roughly, } T \geq \frac{1}{2\gamma_{lb}^2} \cdot \frac{1}{\varepsilon} \right) \end{array}$$

Remark. *These bounds are: for MajBoost from Corollary 2.3 of [Fre95], for AdaBoost from (21) and (22) of [FS97], and for MadaBoost from Theorem 2 of [DW00a]. Theorem 2 of [DW00a] proves the bound for a slightly modified version of MadaBoost, but here we ignore this difference.*

Notice that the convergence time of MadaBoost is exponentially larger than that of MajBoost and AdaBoost. Although we have not been able to prove a better bound, our experiments show that there is not so much difference in convergence speed between MadaBoost and AdaBoost, and we believe that MadaBoost has more or less similar convergence speed. In fact, we can show that this holds for first fixed number of steps. A similar convergence speed is also guaranteed if we ignore examples that are misclassified many times by previously obtained weak hypotheses.

As a final remark on boosting, we mention methods or frameworks for implementing boosting techniques. There are two frameworks:- *boosting by sub-sampling* and *boosting by filtering* [Fre95]¹. The difference between these two frameworks is simply a way to define the “training set”, the set S used in the procedure of Figure 1. In the sub-sampling framework, S is a set of a certain number m of examples chosen from X by using the original example generator \mathbf{EX}_X . On the other hand, in the filtering framework, S is simply the original set X . The distribution D that is given first is defined accordingly. That is, in the sub-sampling framework, D is the uniform distribution on S , while it is the original distribution D_X assumed on X in the filtering framework. (In our example task, the assumed distribution D_X is just the uniform distribution on X .)

Now we have gone through (though quickly) basics of boosting techniques. Are we ready to write a program for solving our example task? Maybe not. What is missing in the above explanation? First of all, we have not explained how to design the weak learner **WeakLearn**. But for weak learners, several weak learning algorithms have been proposed, from a simple decision stump selection to an advanced decision tree constructor such as C4.5. Here we leave this issue to the other papers (see, e.g., [Qui96, Die98]) and assume that some reasonable weak learner is available. Then are we ready now? Still not. We still have to determine some parameters that are important for the performance of an obtained algorithm, and setting these parameters appropriately is not a trivial task at all!

Take MajBoost as our first choice, and consider the implementation of MajBoost in the sub-sampling framework. Then we will soon face the parameter setting problem; we have to determine the parameter γ , which should be a lower bound of advantages $\gamma_1, \dots, \gamma_T$ of obtained weak hypotheses; otherwise, the boosting mechanism does not seem to work (at least, there is no theoretical justification). On the other hand, since T is roughly proportional to $1/\gamma^2$, if we chose γ too small, the total boosting rounds would become unnecessarily large. This problem is solved in the other two boosting techniques, i.e., AdaBoost and MadaBoost. In these techniques, weights are determined appropriately depending on the advantages of (so far) obtained hypotheses. That is, they enable us to design an algorithm that is adaptive to the advantages of weak hypotheses.

Note, however, that we have not yet completely solved the parameter setting problem. When we apply, e.g., AdaBoost to design an algorithm in the sub-sampling framework, we still have to

¹These frameworks are also sometimes refer as *boosting by re-weighting* and *boosting by re-sampling*.

determine the total number T of boosting rounds and the sample size m . It seems that setting these parameters is still at the art level. It would be nice if we can solve such parameter setting problem *algorithmically*. In the following sections, we will show some simple approaches solving this problem.

3 First Approach: Doubling and Cross Validation

As pointed out in the previous section, we would like to determine the total boosting round T and the size m of the training set S . Note first that we do not have to fix T in advance if we terminate the boosting round based on the quality of the obtained combined hypothesis f_t . In fact, as we will see below, it is enough if the error probability of f_t on S under D becomes less than $\varepsilon_0/2$ (provided m is large enough). Thus, let us use this as our stopping condition. Let **AB** denote the algorithm implementing AdaBoost in the sub-sampling framework with this stopping condition. (We assume that **WeakLearn** is appropriately designed. On the other hand, the parameter m is not still fixed in **AB**.) It follows from the bound in Theorem 1, the total boosting steps, which we denote by T_* , is bounded by $\mathcal{O}((1/\gamma_{ib}^2) \ln(1/\varepsilon_0))$.

For the choice of m , we, of course, has some theoretical formula, which is derived by the standard Occam razor type argument based on the estimation of the descriptive complexity of composed hypothesis f_{T_*} . We recall it below. (In the following, we use ℓ_{wk} to denote the descriptive complexity of weak hypotheses that **WeakLearn** produces, which is, more concretely, defined as $\log |H_{\text{wk}}|$, where H_{wk} is the set of all possible weak hypotheses.)

Theorem 2 **AB** yields a hypothesis satisfying our learning goal if m is chosen to satisfy $m \geq (c/\varepsilon_0) \cdot (\ell_{\text{wk}} T_* \log T_*)$, for some constant c .

Remark. This bound is based on the generalization error bound derived from the estimation given in Theorem 8 of [FS97] by using the Chernoff bound. In the literature, on the other hand, a similar generalization error bound is given that is derived by using the Hoeffding bound (which is also known as the additive Chernoff bound). But the sample size estimated by using the latter bound becomes $\mathcal{O}((1/\varepsilon_0^2) \cdot (\ell_{\text{wk}} T_* \log T_*))$.

Thus, we roughly have $m = \tilde{\mathcal{O}}(\ell_{\text{wk}} T / \varepsilon_0)$. (By $\tilde{\mathcal{O}}$ notation, we mean to ignore some logarithmic factors.) Then from the bound for T_* , we could estimate the sample size $m = \tilde{\mathcal{O}}(\ell_{\text{wk}} / (\varepsilon_0 \gamma_{ib}^2))$. But this sample size is essentially the same as the one that MajBoost needs in the sub-sampling framework! The advantage of “adaptivity” of AdaBoost is lost at this point.

We can get around this problem by using some simple algorithmic tricks². Instead of fixing m in advance, we execute **AB** several times by changing m ; starting from a certain small value, e.g., $m = 20$, we double m at each execution. The crucial point of using this “doubling trick” is the existence of some reliable termination test. That is, it should be possible to determine whether

²Since both “doubling” and “cross validation” techniques are now standard, the idea explained here may not be new. In fact, in the paper [FS97] introducing AdaBoost, the “cross validation” technique is suggested.

```

Procedure FiltEX $_{D_t}$ ;
begin
  repeat
    use EX $_X$  to generate an example  $(x, b)$ ;
     $p(x) \leftarrow w_{t-1}(x)/D_X(x)$ ;
    accept  $(x, b)$  with probability  $p(x)$ ;
  until some example  $(x, b)$  is accepted;
  return  $(x, b)$ ;
end-procedure.

```

Figure 2: An example generator **FiltEX** $_{D_t}$ for the filtering framework.

m exceeds the size that is necessary and sufficient for **AB** to produce a desired hypothesis. Here we use the standard “cross validation” technique. That is, we use **EX** $_X$ to generate examples of X and check the generalization error of a hypothesis that **AB** yields under the current choice of m . By using the Chernoff bound, we can show that $\tilde{O}(1/\varepsilon_0)$ examples are enough for this cross validation test. (Some small logarithmic factor is necessary to bound the *total* error probability. In the following, the \tilde{O} notation will be introduced time to time, due to the same reason.)

Now let **AB** $_{CV}$ be an algorithm implementing this idea. Also let m_* denote the sample size that is necessary and sufficient for our task. Then we can show the following bound for **AB** $_{CV}$.

Theorem 3 *With high probability, **AB** $_{CV}$ terminates and yields a hypothesis satisfying our learning goal by executing **AB** for $\lceil \log m_* \rceil$ times. In this case, the number of examples needed is at most $4m_*$ (for the executions of **AB**) plus $\tilde{O}(1/\varepsilon_0)$ (for the cross validation).*

So far, we have discussed about the sample complexity. But it is easy to see that in the execution of **AB**, the time needed for each boosting round is roughly $\tilde{O}(m)$ except for the time needed for the weak learner **WeakLearn**. Thus, if the computation time of **WeakLearn** is also linear (or, almost linear) to the sample size m , we may assume that the actual computation time of **AB** and **AB** $_{CV}$ is proportional to the number of examples used.

4 Second Approach: Filtering Framework

A good news is that in the filtering framework, we do not have to worry about the parameters m and T . In the filtering framework, S is just X , and, as we will see below, the termination (of boosting rounds) can be tested in terms of the quality of obtained combined hypothesis on X . In a sense, we can incorporate the cross validation into the boosting process. Unfortunately, however, AdaBoost cannot be implemented in the filtering framework, and this was why we proposed MadaBoost in [DW00a].

In the filtering framework, examples are taken from X directly, and it is necessary to generate examples under modified distributions D_1, D_2, \dots . But we are provided only **EX** $_X$, the procedure

generating examples of X under the original (which is uniform in our case) distribution D_X . Thus, we need some procedure generating examples from X under a given distribution D_t . Such a procedure is given by Freund in [Fre95]. Let us first recall this procedure. Figure 2 states the outline of the procedure, which is denoted as **FiltEX** $_{D_t}$, that generates examples under the distribution D_t that is used at the t th boosting round. The function w_{t-1} is the one defined in Section 2 for each boosting technique. Recall that D in the definition of w_{t-1} is D_X in the filtering framework.

The probability that each (x, b) is generated at one repeat-iteration of **FiltEX** $_{D_t}$ is $p(x)D_X(x) = w_{t-1}(x)$. Hence, the probability that some $x \in X$ is generated at one repeat-iteration is $W_{t-1} = \sum_{x \in X} w_{t-1}(x)$. Thus, for generating one example by **FiltEX** $_{D_t}$, its repeat-iteration (hence, **EX** $_X$) is executed *on average* W_{t-1} times. Then if W_{t-1} is small, the cost of generating one example by **FiltEX** $_{D_t}$ becomes large. Fortunately, however, from the following relation for MajBoost and MadaBoost, we can prove that W_{t-1} cannot be so small.

Theorem 4 *Under the definition of w_{t-1} of MajBoost (resp., MadaBoost), it holds that $D_X(\{f_{t-1}(x) \neq l_*(x)\}) \leq W_{t-1}$.*

Remark. *For AdaBoost, this bound does not hold, which is due to its exponentially increasing weight function.*

It follows from this theorem that if the current composed hypothesis f_{t-1} is not accurate enough (more specifically, if its error probability is larger than ε_0), then the probability that some example is generated at one repeat-iteration of **FiltEX** $_{D_t}$ is at least ε_0 ; hence, with high probability, **FiltEX** $_{D_t}$ generates one example by executing **EX** $_X$ at most c/ε_0 times. Therefore, if **FiltEX** $_{D_t}$ does not terminate after executing **EX** $_X$ $\tilde{\mathcal{O}}(1/\varepsilon_0)$ times, the boosting algorithm can safely yield the current composed hypothesis f_{t-1} as an answer. That is, the termination test of the whole boosting procedure is included in this generation procedure.

There is one more point that we need to consider for the filtering framework. In the t th boosting round, we assume that the weak learner **WeakLearn** produces some weak hypothesis h_t with advantage γ_t . But since the weak learner is executed on X , it again should use some set of data sampled from X under the distribution D_t . We should consider some way to implement this part. Notice here that $\mathcal{O}(1/\gamma_t^2)$ examples is necessary and sufficient to guarantee that the obtained h_t indeed has advantage approximately γ_t .

If we know that some lower bound γ_{lb} of γ_t , then there is an easy implementation. Just generate $\mathcal{O}(1/\gamma_{lb}^2)$ examples by using **FiltEX** $_{D_t}$, and pass them to the weak learner. Then we can show, by using the Hoeffding bound, that the hypothesis h_t that the weak learner produces has advantage approximately γ_t with high probability. (Of course, we assume here the correctness of the weak learner.) But we would like to obtain such a hypothesis without knowing γ_{lb} in advance. This can be achieved by our adaptive sampling algorithms [DGW01].

The basic idea is simple and in fact the same as before. We use “doubling” and “cross validation”. To be concrete, let us consider here the problem of estimating the advantage γ_* of a given weak hypothesis h (that is obtained at some boosting round). We estimate the advantage

of h by computing its advantage on generated examples, where the number of examples is determined by the current guess γ of γ_* . We start with $\gamma = 1/2$, that is, guessing h has the maximum advantage, and reduce γ by half each time the estimation fails. At each estimation, we generate examples and estimate γ_* on these examples; let $\hat{\gamma}$ be the obtained estimation. The number of examples is large enough so that γ_* is in the range $[\hat{\gamma} - \gamma/4, \hat{\gamma} + \gamma/4]$ with high probability. (By using the Hoeffding bound, we can show that $\mathcal{O}(1/\gamma^2)$ examples are enough.) This estimation fails if $\hat{\gamma} < \gamma/2$, because then our current guess γ of γ_* is too large. Otherwise, we can conclude that $\hat{\gamma} - \gamma/4$ is a reasonable estimation of γ_* . It is easy to see that the estimation computed in this way is within the range $[\gamma_*/2, \gamma_*]$ with high probability. Also the total number of examples is $\mathcal{O}(1/\gamma_*^2)$, which is of the same order compared with the case when we *know* γ_* in advance.

The approach is essentially the same for computing a weak hypothesis by using any reasonable weak learning algorithm. Following this approach, we can obtain a nearly best weak hypothesis h_t at the t th boosting round by using $\tilde{\mathcal{O}}(1/\gamma_t^2)$ examples. In [DGW01], a slightly more sophisticated algorithm for a simple hypothesis selection is presented and its performance is analyzed more carefully.

Now, following the ideas explained above, we can define two algorithms $\mathbf{MjB}_{\text{Filt}}$ and $\mathbf{MaB}_{\text{Filt}}$ that implement respectively MajBoost and MadaBoost in the filtering framework.

Finally, let us estimate the efficiency of these algorithms $\mathbf{MjB}_{\text{Filt}}$ and $\mathbf{MaB}_{\text{Filt}}$. Here we estimate the number m_* of executions of \mathbf{EX}_X , because the total computation time is almost proportional to m_* (provided $\mathbf{WeakLearn}$ is almost linear time). As before, we use T_* to denote the total number of boosting rounds executed. By summarizing our discussion, we can show the following bound.

Theorem 5 *With high probability, $\mathbf{MjB}_{\text{Filt}}$ and $\mathbf{MaB}_{\text{Filt}}$ respectively terminates and yields a hypothesis satisfying our learning goal by executing $\tilde{\mathbf{EX}}_X$ at most $\tilde{\mathcal{O}}(1/\varepsilon_0) \cdot \tilde{\mathcal{O}}(\sum_{1 \leq i \leq T_*} 1/\gamma_i^2)$ times. This bound is, roughly, $\tilde{\mathcal{O}}(T_*/(\varepsilon_0 \gamma_{lb}^2))$.*

5 Concluding Remarks

We have shown two algorithmic approaches for implementing boosting techniques to solve a simple classification problem on a huge dataset. Since these implementations are adaptive, it is hard to compare their efficiency analytically. Here we *very roughly*³ assume that the boosting steps T_* is more or less the same for any of three boosting algorithms, and compare their running time. Then provided that $\mathbf{WeakLearn}$ runs in almost linear time, we have the following time complexity: \mathbf{AB}_{CV} $\tilde{\mathcal{O}}(T_*/\varepsilon_0)$, $\mathbf{MjB}_{\text{Filt}}$ and $\mathbf{MaB}_{\text{Filt}}$ $\tilde{\mathcal{O}}((1/\varepsilon_0) \cdot (\sum_{1 \leq i \leq T_*} 1/\gamma_i^2))$, which is roughly $\tilde{\mathcal{O}}((T_*/\varepsilon_0) \cdot (1/\gamma_{lb}^2))$. Thus, the implementation by the filtering framework is slower by the factor

³In fact, this is a quite rough comparison. It seems that $\mathbf{MjB}_{\text{Filt}}$ that requires to set the parameter γ by an appropriate γ_{lb} is slow, though again we may use the doubling trick. On the other hand, while it seems that MadaBoost has convergence speed similar to AdaBoost, it has not been proved yet.

of $1/\gamma_{lb}^2$. The situation, however, would change if we want to use a bit more complicated weak learner for **WeakLearn**. Usually a weak learning algorithm, in the sub-sampling framework, needs to use the whole m_* examples in S to obtain a weak hypothesis. Thus, if it has a quadratic running time, then the running time of **AB_{CV}** becomes $\tilde{O}(m_*^2)$, which is $\tilde{O}(T_*^2/\varepsilon_0^2)$. On the other hand, the number of examples passed to **WeakLearn** in **MjB_{Filt}** and **MaB_{Filt}** is $\tilde{O}(1/\gamma_t^2)$ at the t th boosting round. Thus, the computation time becomes $\tilde{O}((1/\varepsilon_0) \cdot (\sum_{1 \leq i \leq T_*} 1/\gamma_i^4))$, and **MaB_{Filt}** may be faster than **AB_{CV}** in this situation. As pointed out, e.g., by Morishita et al [Mor00], a stronger weak learner sometimes yields a better learning algorithm, and in general the filtering framework is better for using costly weak learning algorithms.

We have explained some algorithmic techniques to apply boosting for huge data classification problems. But in practice, almost all example sets have some amount of errors, and the proportion of error may become even larger if a dataset becomes large. On the other hand, it has been criticized that boosting techniques, e.g., AdaBoost, are too sensitive to erroneous examples. One interesting idea is to ignore examples that are misclassified by previous weak hypotheses more than a certain number of times. But then we will have another parameter setting problem; that is, determining this threshold. An algorithmic approach to this parameter setting problem would be an important and interesting future problem.

References

- [CSS00] M. Collins, R.E. Schapire, and Y. Singer, Logistic regression, AdaBoost and Bregman Distance, in *Proc. of the Thirteenth Annual ACM Workshop on Computational Learning Theory (COLT'00)*, ACM, 158–169, 2000.
- [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.
- [DW00a] C. Domingo and O. Watanabe, MadaBoost: A modification of AdaBoost, in *Proc. of the Thirteenth Annual ACM Workshop on Computational Learning Theory (COLT'00)*, ACM, 180–189, 2000.
- [DW00b] C. Domingo and O. Watanabe, Scaling up a boosting-based learner via adaptive sampling, in *Proc. of Knowledge Discovery and Data Mining (PAKDD'00)*, Lecture Notes in AI 1805, 317–328 (2000).
- [DGW01] C. Domingo, R. Gavaldà, and O. Watanabe, Adaptive sampling methods for scaling up knowledge discovery algorithms, *Data Mining and Knowledge Discovery* (special issue edited by H. Liu and H. Motoda), 2001, to appear.
- [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 (COLT'99)*, ACM, 102–113, 1999.
- [FHT98] J. Friedman, T. Hastie, and R. Tibshirani, Additive logistic regression: a statistical view of boosting, Technical report, 1998.
- [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.
- [Mor00] S. Morishita, a personal communication.
- [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.
- [Wat00a] O. Watanabe, Simple sampling techniques for discovery science, *IEICE Transactions on Information and Systems*, E83-D(1), 19–26, 2000.
- [Wat00b] O. Watanabe, Sequential sampling techniques for algorithmic learning theory, in *Proc. 11th Int'l Conference on Algorithmic Learning Theory (ALT'00)*, Lecture Notes in AI, xx–xx, 2000.