AveBoost2: Boosting for Noisy Data

Nikunj C. Oza

Computational Sciences Division
NASA Ames Research Center
Mail Stop 269-1
Moffett Field, CA 94035-1000, USA
oza@email.arc.nasa.gov

Abstract. AdaBoost [4] is a well-known ensemble learning algorithm that constructs its *base* models in sequence. AdaBoost constructs a distribution over the training examples to create each base model. This distribution, represented as a vector, is constructed with the goal of making the next base model's mistakes uncorrelated with those of the previous base model [5]. We previously [7] developed an algorithm, AveBoost, that first constructed a distribution the same way as AdaBoost but then averaged it with the previous models' distributions to create the next base model's distribution. Our experiments demonstrated the superior accuracy of this approach. In this paper, we slightly revise our algorithm to obtain non-trivial theoretical results: bounds on the training error and generalization error (difference between training and test error). Our averaging process has a regularizing effect which leads us to a worse training error bound for our algorithm than for AdaBoost but a better generalization error bound. This leads us to suspect that our new algorithm works better than AdaBoost on noisy data. For this paper, we experimented with the data that we used in [7] both as originally supplied and with added label noise—some of the data has its original label changed randomly. Our algorithm's experimental performance improvement over AdaBoost is even greater on the noisy data than the original data.

1 Introduction

AdaBoost [4] is one of the most well-known and highest-performing ensemble classifier learning algorithms [3]. It constructs a sequence of base models, where each model is constructed based on the performance of the previous model on the training set. In particular, AdaBoost calls the base model learning algorithm with a training set weighted by a distribution.[1] After the base model is created, it is tested on the training set to see how well it learned. We assume that the base model learning algorithm is a *weak learning algorithm*; that is, with high probability, it produces a model whose probability of misclassifying an example

[1] If the base model learning algorithm cannot take a weighted training set as input, then one can create a sample with replacement from the original training set according to the distribution and call the algorithm with that sample.

is less than 0.5 when that example is drawn from the same distribution that generated the training set. The point is that such a model performs better than random guessing.[2] The weights of the correctly classified examples and misclassified examples are scaled down and up, respectively, so that the two groups' total weights are 0.5 each. The next base model is generated by calling the learning algorithm with this new weight distribution and the training set. The idea is that, because of the weak learning assumption, at least some of the previously misclassified examples will be correctly classified by the new base model. Previously misclassified examples are more likely to be classified correctly because of their higher weights, which focus more attention on them. AdaBoost scales the distribution with the goal of making the next base model's mistakes uncorrelated with those of the previous base model [5].

AdaBoost is notorious for performing poorly on noisy datasets [3], such as those with label noise—that is, some examples were randomly assigned the wrong class label. Because these examples are inconsistent with the majority of examples, they tend to be harder for the base model learning algorithm to learn. AdaBoost increases the weights of examples that the base model learning algorithm did not learn correctly. Noisy examples are likely to be incorrectly learned by many of the base models so that eventually these examples' weights will dominate those of the remaining examples. This causes AdaBoost to focus too much on the noisy examples at the expense of the majority of the training examples, leading to poor performance on new examples.

We previously [7] presented an algorithm, called AveBoost, which calculates the next base model's distribution by first calculating a distribution the same way as in AdaBoost, but then averaging it elementwise with those calculated for the previous base models. This averaging mitigates AdaBoost's tendency to increase the weights of noisy examples to excess. In our previous work we presented promising experimental results. However, we did not present theoretical results. In our subsequent research, we were unable to derive a non-trivial training error bound for the algorithm presented in [7]. In this paper, we present a slight modification to AveBoost which allows us to obtain both a non-trivial training error bound and a generalization error bound (difference between training error and test error). We call this algorithm AveBoost2. In Section 2, we review AdaBoost. In Section 3, we describe the AveBoost2 algorithm and state how it is different from AveBoost. In Section 4, we present our training error bound and generalization error bound. The averaging in our algorithm has a regularizing effect; therefore, as expected, our training error bound is worse than that of AdaBoost but our generalization error bound is better than AdaBoost's. In Section 5, we present an experimental comparison of our new AveBoost2 with AdaBoost on some UCI datasets [1] both in original form and with 10% label noise added. Section 6 summarizes this paper and describes ongoing and future work.

[2] The version of AdaBoost that we use was designed for two-class classification problems. However, it is often used for a larger number of classes when the base model learning algorithm is strong enough to have an error less than 0.5 in spite of the larger number of classes.

```
**AdaBoost**$(\{(x_1, y_1), \ldots, (x_m, y_m)\}, L_b, T)$
    Initialize $d_{1,i} = 1/m$ for all $i \in \{1, 2, \ldots, m\}$.
    For $t = 1, 2, \ldots, T$:
        $h_t = L_b(\{(x_1, y_1), \ldots, (x_m, y_m)\}, \mathbf{d}_t)$
        Calculate the error of $h_t : \epsilon_t = \sum_{i:h_t(x_i) \neq y_i} d_{t,i}$.
        If $\epsilon_t \geq 1/2$ then,
            set $T = t - 1$ and abort this loop.
        $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
        Calculate distribution $\mathbf{d}_{t+1}$:
            $w_i = d_{t,i} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
            $d_{t+1,i} = \frac{w_i}{\sum_{i=1}^{m} w_i}$
    **Output** the final hypothesis:
        $h_{fin}(x) = \text{argmax}_{y \in Y} \sum_{t:h_t(x)=y} log \frac{1}{\beta_t}$
```

Fig. 1. AdaBoost algorithm: $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ is the training set, L_b is the base model learning algorithm, and T is the maximum allowed number of base models.

2 AdaBoost

Figure 1 shows AdaBoost's pseudocode. AdaBoost constructs a sequence of base models h_t for $t \in \{1, 2, \ldots, T\}$, where each model is constructed based on the performance of the previous base model on the training set. In particular, AdaBoost maintains a distribution over the m training examples. The distribution $\mathbf{d}_1$ used in creating the first base model gives equal weight to each example ($d_{1,i} = 1/m$ for all $i \in \{1, 2, \ldots, m\}$). AdaBoost now enters the loop, where the base model learning algorithm L_b is called with the training set and $\mathbf{d}_1$. The returned model h_1 is then tested on the training set to see how well it learned. The total weight of the misclassified examples (ϵ_1) is calculated. The weights of the correctly-classified examples are multiplied by $\epsilon_1/(1 - \epsilon_1)$ so that they have the same total weight as the misclassified examples. The weights of all the examples are then normalized so that they sum to 1 instead of $2\epsilon_1$. AdaBoost assumes that L_b is a *weak learner*, i.e., $\epsilon_t < \frac{1}{2}$ with high probability. Under this assumption, the total weight of the misclassified examples $\epsilon_t < 1/2$ is increased to $1/2$ and the total weight of the correctly classified examples $1 - \epsilon_t > 1/2$ is decreased to $1/2$. This is done so that, by the weak learning assumption, the next model h_{t+1} will classify at least some of the previously misclassified examples correctly. Returning to the algorithm, the loop continues, creating the T base models in the ensemble. The final ensemble returns, for a new example, the one class in the set of classes Y that gets the highest weighted vote from the base models. Each base model's vote is proportional to its accuracy on the weighted training set used to train it.

```
**AveBoost2**$(\{(x_1, y_1), \ldots, (x_m, y_m)\}, L_b, T)$
    Initialize $d_{1,i} = 1/m$ for all $i \in \{1, 2, \ldots, m\}$.
    For $t = 1, 2, \ldots, T$:
        $h_t = L_b(\{(x_1, y_1), \ldots, (x_m, y_m)\}, \mathbf{d}_t)$
        Calculate the error of $h_t : \epsilon_t = \sum_{i:h_t(x_i) \neq y_i} d_{t,i}$.
        If $\epsilon_t \geq 1/2$ then,
            set $T = t - 1$ and abort this loop.
        $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
        $\gamma_t = \frac{2(1-\epsilon_t)t+1}{2\epsilon_t t+1}$
        Calculate distribution $\mathbf{d}_{t+1}$:
            For $i = 1, 2, \ldots, m$:
                $w_i = d_{t,i} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
                $c_{t,i} = \frac{w_i}{\sum_{i=1}^{m} w_i}$
                $d_{t+1,i} = \frac{t d_{t,i} + c_{t,i}}{t+1}$
    **Output** the final hypothesis:
        $h_{fin}(x) = \text{argmax}_{y \in Y} \sum_{t:h_t(x)=y} log \frac{1}{\beta_t \gamma_t}$
```

Fig. 2. AveBoost2 algorithm: $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ is the training set, L_b is the base model learning algorithm, and T is the maximum allowed number of base models.

3 AveBoost2 algorithm

Figure 2 shows our new algorithm, AveBoost2. Just as in AdaBoost, AveBoost2 initializes $d_{1,i} = 1/m$ for all $i \in \{1, 2, \ldots, m\}$. Then it goes inside the loop, where it calls the base model learning algorithm L_b with the training set and distribution $\mathbf{d}_1$ and calculates the error ϵ_1 of the resulting base model h_1. It then calculates $\mathbf{c}_1$, which is the distribution that AdaBoost would use to construct the next base model. However, AveBoost2 averages this with $\mathbf{d}_1$ to get $\mathbf{d}_2$, and uses this $\mathbf{d}_2$ instead. Showing that the $\mathbf{d}_t$'s in AveBoost2 are distributions is a trivial proof by induction. For the base case, $\mathbf{d}_1$ is constructed to be a distribution. For the inductive part, if $\mathbf{d}_t$ is a distribution, then $\mathbf{d}_{t+1}$ is a distribution because it is a convex combination of $\mathbf{d}_t$ and $\mathbf{c}_t$, both of which are distributions. The vector $\mathbf{d}_{t+1}$ is a running average of $\mathbf{d}_1$ and the vectors $\mathbf{c}_\mathbf{q}$ for $q \in \{1, 2, \ldots, t\}$.

Returning to the algorithm, the loop continues for a total of T iterations. Then the base models are combined using a weighted voting scheme slightly different from that of AdaBoost and the original AveBoost from [7]: each model's weight is $log(1/(\beta_t\gamma_t))$ instead of $log(1/\beta_t)$. AveBoost2 is actually AdaBoost with β_t replaced by $\beta_t\gamma_t$. However, we wrote the AveBoost2 pseudocode as we did to make the running average calculation of the distribution explicit.

AveBoost2 can be seen as a relaxed version of AdaBoost. When training examples are noisy and therefore difficult to fit, AdaBoost is known to increase the weights of those examples to excess and overfit them [3] because many consecutive base models may not learn them properly. AveBoost2's averaging does not allow the weights of noisy examples to increase rapidly, thereby mitigating the overfitting problem. We therefore expect AveBoost2 to outperform AdaBoost on the noisy datasets to a greater extent than on the original datasets. We also expect AveBoost2's advantage to be greater for smaller numbers of base models. When AveBoost2 creates a large ensemble, later training set distributions (and therefore later base models) cannot be too different from each other because they are prepared by averaging over many previous distributions. Therefore, we expect that later models will not have as much of an impact on performance.

4 Theory

In this section, we give bounds on the training error and generalization error (difference between training and test error). Not surprisingly, the relaxed nature of AveBoost2 relative to AdaBoost caused us to obtain a worse training error bound but superior generalization error bound for AveBoost2 relative to AdaBoost. Due to space limitations, we defer the proofs and more intuition on the theoretical frameworks that we use to a longer version of this paper. AveBoost2's training error bound is stated in the following theorem.

Theorem 1. *In AveBoost2, suppose the weak learning algorithm L_b generates hypotheses with errors $\epsilon_1, \epsilon_2, \ldots, \epsilon_T$ where each $\epsilon_t < 1/2$. Then the ensemble's error $\epsilon = \sum_{i:h_{fin}(x_i) \neq y_i} d_{1,i}$ is bounded as follows:*

$$\epsilon \leq \prod_{t=1}^{T} \frac{t+1}{\sqrt{t^2 + \frac{t}{2\epsilon_t(1-\epsilon_t)} + \frac{1}{4\epsilon_t(1-\epsilon_t)}}}.$$

This bound is non-trivial ($\epsilon \leq 1$); but greater than that of AdaBoost [4]:

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}.$$

To derive our generalization error bound, we use the algorithmic stability framework of [6]. Intuitively, algorithmic stability is similar to Breiman's notion of stability [2]—the more stable a learning algorithm is, the less of an effect changes to the training set have on the model returned. The more stable the learning algorithm is, the smaller the difference between the training and test errors tends to be, assuming that the training and test sets are drawn from the same distribution. We show that AveBoost2 is more stable than AdaBoost; therefore, the difference between the training and test errors is lower. We first give some preliminaries from [6] and then state our new result.

For the following, $\mathcal{X}$ is the space of possible inputs, $\mathcal{Y} = \{0, 1\}$ is the set of possible labels, and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.

Definition 1 (Definition 2.5 from [6]). *A* learning algorithm *is a process which takes as input a distribution p on $\mathcal{Z}$ with finite support and outputs a function $f_p : \mathcal{X} \to [0, 1]$. For $S \in \mathcal{Z}^m$ for some positive integer m, f_S means f_p where p is the uniform distribution on S.*

In the following, the error of f on an example (x, y) is $c(f, (x, y)) = |f(x) - y|$.

Definition 2 (Definition 2.11 from [6]). *A learning algorithm has L_1-stability λ if, for any two distributions p and q on $\mathcal{X}$ with finite support,*

$$\forall z \in \mathcal{Z}, |c(f_p, z) - c(f_q, z)| \leq \lambda \|p - q\|_1.$$

In the following, D is a distribution on $\mathcal{Z}$, $S \sim D^m$ is a set of m examples drawn from $\mathcal{Z}$ according to D, and $S^{i,u}$ is S with example $i \in \{1, 2, \ldots, m\}$ removed (each i is chosen with probability $1/m$) and example $u \sim D$ added.

Definition 3 (Definition 2.14 from [6]). *A learning algorithm is (β, δ)-stable if*

$$P_{S \sim D^m}(|c(f_S, z) - c(f_{S^{i,u}}, z)| \leq \beta) \geq 1 - \delta.$$

Intuitively, f_S and $f_{S^{i,u}}$ are models that result from running the learning algorithm on two slightly different training sets. As β and δ decrease, the probability of having smaller differences in errors between these two models increases, which means that the learning algorithm is more stable. Greater stability implies lower generalization error according to the following theorem. In the following, $Err_S(f_S)$ is the training error (error on the training set S) and $Err_D(f_S)$ is the test error, i.e., the error on an example (x, y) chosen at random according to distribution D.

Theorem 2 (Theorem 3.4 from [6]). *Suppose a (β, δ)-stable learning algorithm returns a hypothesis f_S for any training set $S \sim D^m$. Then for all $\tau > 0$ and $m \geq 1$,*

$$P_{S \sim D^m}(|Err_S(f_S) - Err_D(f_S)| > \tau + \beta + \delta) \leq 2exp\left(\frac{-\tau^2 m}{2(2m\beta + 1)^2}\right) + \frac{4m^2\delta}{2m\beta + 1}.$$

Intuitively, this theorem shows that lower values of β and δ lead to lower probabilities of large differences between the training and test errors. We can finally state our theorem on AveBoost2's stability.

Theorem 3. *Suppose the base model learning algorithm has L_1-stability λ and $\min_{t \in \{1,2,\ldots,T\}} \epsilon_t > \epsilon_* > 0$. Then, for sufficiently large m and for all T, AveBoost2 is (β, δ)-stable, where*

Table 1. The datasets used in the experiments

Data Set	Training Set	Test Set	Inputs	Classes
Promoters	84	22	57	2
Balance	500	125	4	3
Breast Cancer	559	140	9	2
German Credit	800	200	20	2
Car Evaluation	1382	346	6	4
Chess	2556	640	36	2
Mushroom	6499	1625	22	2
Nursery	10368	2592	8	5
Connect4	54045	13512	42	3

$$\beta = \frac{2}{m}\sum_{t=1}^{T} 2^{3t-2}\left(\frac{(\lambda+1)t}{\epsilon_*}\right)^t$$

$$\delta = exp\left(\frac{-m\epsilon_*^2}{2}\right).$$

For AdaBoost, the theorem is the same except that [6]

$$\beta = \frac{2}{m}\sum_{t=1}^{T}\frac{2^{t^2+1}(\lambda+1)^t}{\epsilon_*^{2t-1}},$$

which is larger than the corresponding β for AveBoost2. This means that AveBoost2's generalization error is less than that of AdaBoost by Theorem 2.

Note that as ϵ_* decreases toward zero, the β and δ for both AdaBoost and AveBoost increase, which means both algorithms are less stable. This makes sense because, as ϵ_* decreases, the upper bounds on the weights assigned to each base model ($log\frac{1}{\beta_t}$ and $log\frac{1}{\beta_t\gamma_t}$) increase. This means that each individual base model's potential influence on the ensemble's result is higher, so that changes in the individual base models lead to greater changes in the ensemble's predictions.

5 Experimental Results

In this section, we compare AdaBoost and AveBoost2 on the nine UCI datasets [1] described in Table 1. We ran both algorithms with three different values of T, which is the maximum number of base models that the algorithm is allowed to construct: 10, 50, and 100. Each result reported is the average over 50 results obtained by performing 10 runs of 5-fold cross-validation. Table 1 shows the sizes of the training and test sets for the cross-validation runs. We also repeated

Table 2. Performance of AveBoost2 compared to AdaBoost

ORIGINAL	Num. Base Models			10% NOISE	Num. Base Models		
Base Model	10	50	100	Base Model	10	50	100
Naive Bayes	+4=4-1	+4=4-1	+4=4-1	Naive Bayes	+8=1-0	+8=1-0	+7=2-0
Decision Trees	+2=6-1	+1=6-2	+2=6-1	Decision Trees	+6=2-1	+5=3-1	+6=2-1
Decision Stumps	+1=6-2	+1=6-2	+1=6-2	Decision Stumps	+0=7-2	+1=7-1	+1=7-1

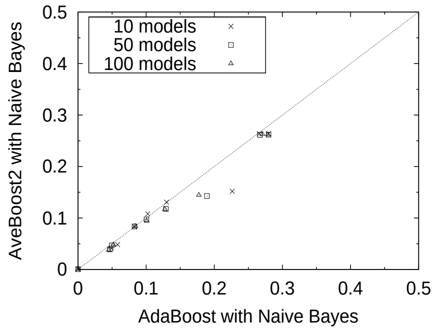

Fig. 3. Test set error rates of AdaBoost vs. AveBoost2 (Naive Bayes, original datasets)

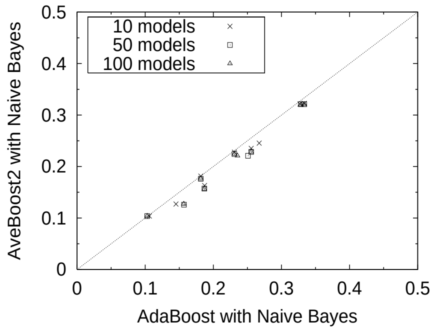

Fig. 4. Test set error rates of AdaBoost vs. AveBoost2 (Naive Bayes, noisy datasets)

these runs after adding 10% label noise. That is, we randomly chose 10% of the examples in each dataset and changed their labels to one of the remaining labels with equal probability.

Table 2 shows how often AveBoost2 significantly outperformed, performed comparably with, and significantly underperformed AdaBoost. For example, on the original datasets and with 10 Naive Bayes base models, AveBoost2 significantly outperformed[3] AdaBoost on four datasets, performed comparably on four datasets, and performed significantly worse on one, which is written as "+4=4-1." Figures 3 and 4 compare the error rates of AdaBoost and AveBoost2 with Naive Bayes base models on the original and noisy datasets, respectively. In all the plots presented in this paper, each point marks the error rates of two algorithms when run with the number of base models indicated in the legend and a particular dataset. The diagonal lines in the plots contain points at which the two algorithms have equal error. Therefore, points below/above the line correspond to the error of the algorithm indicated on the y-axis being less than/greater than the error of the algorithm indicated on the x-axis, respectively. For Naive Bayes base models, AveBoost2 performs much better than AdaBoost overall, especially on the noisy datasets.

[3] We use a t-test with $\alpha = 0.05$ to compare all the classifiers in this paper.

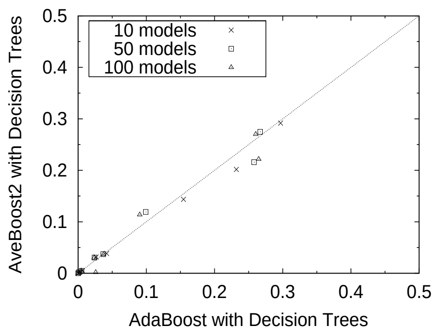

Fig. 5. Test set error rates of AdaBoost vs. AveBoost2 (Decision Trees, original datasets)

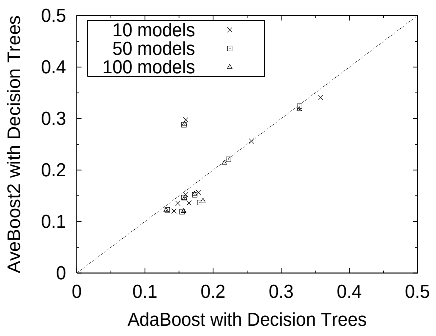

Fig. 6. Test set error rates of AdaBoost vs. AveBoost2 (Decision Trees, noisy datasets)

We compare AdaBoost and AveBoost2 using decision tree base models in figure 5 (original datasets) and figure 6 (noisy datasets). On the original datasets, the performances of the two algorithms are comparable. However, on the noisy datasets, AveBoost2 is superior for all except the Balance dataset. On the Balance dataset, AdaBoost actually performed as much as 10% better on the noisy data than the original data, which is strange, and needs to be investigated further. On the other hand, AveBoost2 performed worse on the noisy Balance data than on the original Balance data. Figure 7 gives the error rate comparison between AdaBoost and AveBoost2 with decision stump base models on the original datasets. Figure 8 gives the same comparison on the noisy datasets. With decision stumps, the two algorithms always seem to perform comparably. We suspect that decision stumps are too stable to allow the different distribution calculation methods of AdaBoost and AveBoost2 to yield significant differences. In all the results, we did not see the hypothesized differences in performance as a function of the number of base models.

6 Conclusions

We presented AveBoost2, a boosting algorithm that trains each base model using a training example weight vector that is based on the performances of all the previous base models rather than just the previous one. We discussed our theoretical results and demonstrated empirical results that are superior overall to AdaBoost; especially on datasets with label noise.

Our theoretical and empirical results do not account for what happens as the amount of noise in the data changes. We plan to derive such results. In a longer version of this paper, we plan to perform a more detailed empirical analysis including the performances of the base models and ensembles on the training and test sets, correlations among the base models, ranges of the weights of regular and noisy examples, etc. In [7], we performed such an analysis to a

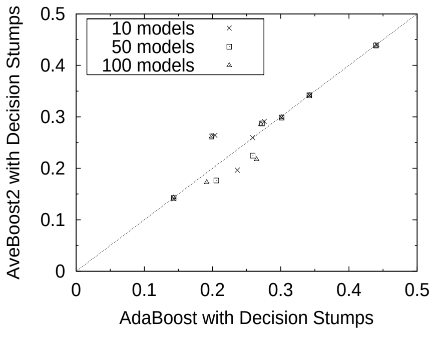

Fig. 7. Test set error rates of AdaBoost vs. AveBoost2 (Decision Stumps, original datasets)

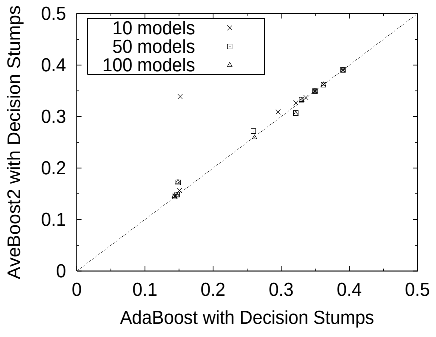

Fig. 8. Test set error rates of AdaBoost vs. AveBoost2 (Decision Stumps, noisy datasets)

limited extent for the original AveBoost and were able to confirm some of what we hypothesized there and in this paper: the base model accuracies tend to be higher than for AdaBoost, the correlations among the base models also tend to be higher, and the ranges of the weights of the training examples tends to be lower. We were unable to repeat this analysis here due to a lack of space. We also plan to compare our algorithm to other efforts to make boosting work better with noisy data, such as identifying examples that have consistently high weight as noisy and; therefore, untrustworthy [8].

References

1. C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1999. (URL: http://www.ics.uci.edu/∼mlearn/MLRepository.html).
2. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
3. Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–158, Aug. 2000.
4. Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996. Morgan Kaufmann.
5. Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 134–144, 1999.
6. Samuel Kutin and Partha Niyogi. The interaction of stability and weakness in adaboost. Technical Report TR-2001-30, University of Chicago, October 2001.
7. Nikunj C. Oza. Boosting with averaged weight vectors. In T. Windeatt and F. Roli, editors, *Proceedings of the Fourth International Workshop on Multiple Classifier Systems*, pages 15–24. Springer, Berlin, 2003.
8. G. Rätsch, T. Onoda, and K.R. Müller. Soft margins for adaboost. *Machine Learning*, 42:287–320, 2001.