Machine Learning: Proceedings of the Fourteenth International Conference, 1997.

Using output codes to boost multiclass learning problems

Robert E. Schapire
AT&T Labs*
600 Mountain Avenue, Room 2A-424
Murray Hill, NJ 07974
schapire@research.att.com

Abstract. This paper describes a new technique for solving multiclass learning problems by combining Freund and Schapire's boosting algorithm with the main ideas of Dietterich and Bakiri's method of error-correcting output codes (ECOC). Boosting is a general method of improving the accuracy of a given base or "weak" learning algorithm. ECOC is a robust method of solving multiclass learning problems by reducing to a sequence of two-class problems. We show that our new hybrid method has advantages of both: Like ECOC, our method only requires that the base learning algorithm work on binary-labeled data. Like boosting, we prove that the method comes with strong theoretical guarantees on the training and generalization error of the final combined hypothesis assuming only that the base learning algorithm perform slightly better than random guessing. Although previous methods were known for boosting multiclass problems, the new method may be significantly faster and require less programming effort in creating the base learning algorithm. We also compare the new algorithm experimentally to other voting methods.

1 INTRODUCTION

Boosting is a general method for improving the accuracy of a learning algorithm. By definition, a boosting algorithm is one which can provably convert any base or "weak" learning algorithm with accuracy just slightly better than random guessing into one with arbitrarily high accuracy. Boosting algorithms work by repeatedly reweighting the examples in the training set and rerunning the weak learning algorithm on these reweighted examples. Boosting effectively forces the weak learning algorithm to concentrate on the hardest examples. Typically, the final combined hypothesis is a weighted vote of the weak hypotheses.

The first boosting algorithms were discovered by Schapire [17] and Freund [6]. Freund and Schapire's most recent boosting algorithm [8], called ADABOOST, has been shown to be very effective in experiments conducted by Drucker and Cortes [5], Jackson and Craven [13], Freund and Schapire [7], Quinlan [15], Breiman [2] and others.

*AT&T Labs is planning to move from Murray Hill. The new address will be: 180 Park Avenue, Florham Park, NJ 07932-0971.

In its simplest form, ADABOOST requires that the accuracy of each weak hypothesis (or classification rule) produced by the weak learner must exceed $1/2$. For binary classification problems (in which each example is labeled by a value in $\{0, 1\}$), this requirement is about as minimal as can be hoped for since random guessing will achieve accuracy $1/2$. However, for multiclass problems in which $k > 2$ labels are possible, accuracy $1/2$ may be much harder to achieve than the random-guessing accuracy rate of $1/k$.

For fairly powerful weak learners, such as decision-tree algorithms, this does not seem to be a problem. Experimentally, C4.5 and CART seem to be capable of producing hypotheses with accuracy $1/2$, even on the difficult distributions of examples produced by boosting [2, 5, 7, 15]. However, the accuracy $1/2$ requirement can often be a difficulty for less powerful weak learners, such as the simple attribute-value tests studied by Holte [12], and used by Jackson and Craven [13] and Freund and Schapire [7] in their boosting experiments. Although overall error rate is often better when more powerful weak learners are used, these less expressive weak learners have the advantage that the final combined hypothesis is usually less complicated, and computation time may be more reasonable, especially for very large datasets.

Freund and Schapire [8] provide one solution to this problem by modifying the form of the weak hypotheses and refining the goal of the weak learner. In this approach, rather than predicting a single class for each example, the weak learner chooses a set of "plausible" labels for each example. For instance, in a character recognition task, the weak hypothesis may predict that a particular example is either a "6," "8" or "9," rather than choosing just a single label. Such a weak hypothesis is then evaluated using a "pseudoloss" measure which, for a given example, penalizes the weak hypothesis for (1) failing to include the correct label in the predicted plausible label set, and (2) for each *in*correct label which *is* included in the plausible set. The final combined hypothesis, for a given example, chooses the single label which occurs most frequently in the plausible label sets chosen by the weak hypotheses (possibly giving more or less weight to some of the weak hypotheses).

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y$
For $t = 1, \ldots, T$:

- Compute distribution D_t over $\{1, \ldots, m\}$.
- Compute coloring $\mu_t : Y \to \{0, 1\}$.
- Train weak learner on examples $(x_1, \mu_t(y_1)), \ldots, (x_m, \mu_t(y_m))$ weighted according to D_t.
- Get weak hypothesis $h_t : X \to \{0, 1\}$.

Compute coefficients $\alpha_1, \ldots, \alpha_T \in \mathbb{R}$.
Output the final hypothesis:

$$H_{\text{final}}(x) = \arg\max_{\ell \in Y} \sum_{t=1}^{T} \alpha_t [\![h_t(x) = \mu_t(\ell)]\!].$$

Figure 1: A generic algorithm combining boosting and ECOC.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y$
Initialize $\tilde{D}_1(i, \ell) = [\![\ell \neq y_i]\!] / (m(k-1))$
/* uniform over all incorrect labels */
For $t = 1, \ldots, T$:

- Train weak learner using pseudoloss defined by $\tilde{D}_t$.
- Get weak hypothesis $\tilde{h}_t : X \to 2^Y$.
- Let
$$\tilde{\epsilon}_t = \tfrac{1}{2} \sum_{i=1}^{m} \sum_{\ell \in Y} \tilde{D}_t(i, \ell) \cdot \left([\![y_i \notin \tilde{h}_t(x_i)]\!] + [\![\ell \in \tilde{h}_t(x_i)]\!] \right)$$
- Let $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \tilde{\epsilon}_t}{\tilde{\epsilon}_t} \right)$.
- Update
$$\tilde{D}_{t+1}(i, \ell) = \frac{\tilde{D}_t(i, \ell) \cdot \exp\left(\alpha_t \left([\![y_i \notin \tilde{h}_t(x_i)]\!] + [\![\ell \in \tilde{h}_t(x_i)]\!] \right) \right)}{Z_t}$$
where Z_t is a normalization factor (chosen so that $\tilde{D}_{t+1}$ will sum to 1).

Output the final hypothesis:

$$H_{\text{final}}(x) = \arg\max_{\ell \in Y} \sum_{t=1}^{T} \alpha_t [\![\ell \in \tilde{h}_t(x)]\!].$$

Figure 2: The pseudoloss-based boosting algorithm AdaBoost.M2.

The exact form of the pseudoloss is under the control of the boosting algorithm, and the weak learning algorithm must therefore be designed to handle changes in the form of the loss measure. This design gives the boosting algorithm the freedom to focus the weak learner not only on the hard to predict *examples*, but also on the *labels* which are hardest to distinguish from the correct label.

This approach works well experimentally [7], but suffers certain drawbacks. First, it requires the design of a weak learner which is responsive to the pseudoloss defined by the boosting algorithm and whose hypotheses generate predictions in the form of plausibility sets. Since most "off-the-shelf" learning algorithms are error-based, this may demand extra effort and creativity on the part of the programmer (and may be completely impossible if the source code for the weak learning algorithm is unavailable).

The second drawback of the pseudoloss approach is that it can be fairly slow. Typically, the running time of the weak learner is $O(k)$ times slower than that of an error-based algorithm for a k-class problem.

In this paper, we describe an alternative method for boosting multiclass learning algorithms. Our method combines boosting with Dietterich and Bakiri's [4] approach based on error-correcting output codes (ECOC), which is designed to handle multiclass problems using only a binary learning algorithm.

Briefly, their approach works as follows: As in boosting, a given "weak" learning algorithm (which need only be designed for two-class problems) is rerun repeatedly. However, unlike boosting, the examples are not reweighted. Instead, on each round, the labels assigned to each example are modified so as to create a new binary labeling of the data which is induced by a simple mapping from the set of labels to $\{0, 1\}$. The sequence of bit assignments for each of the k labels can then be viewed as a "code word." A given test example is then classified by choosing the label whose associated code word is closest in Hamming distance to the sequence of predictions generated by the weak hypotheses. This coding-theoretic interpretation led Dietterich and Bakiri to the beautiful idea of choosing code words with strong error-correcting properties.

The algorithm presented in this paper is a hybrid of the boosting and ECOC approaches. As in boosting, on each round of rerunning the weak learner, the examples are reweighted in a manner focusing on the hardest examples. Then, as in ECOC, the labels are modified to create a binary classification problem. The result is an algorithm that combines the benefits of both approaches: As in ECOC, the weak learning algorithm need only be able to handle binary problems, and with respect to ordinary error rather than the more complicated and time-consuming pseudoloss. Like boosting, the algorithm comes with a strong theoretical guarantee, namely, that if the weak learner can consistently generate weak hypotheses that are slightly better than random guessing (with respect to the distribution and binary example labeling on which it was trained), then the error of the final combined hypothesis can be made arbitrarily small. This is the main theoretical result of this paper.

In the rest of the paper, we describe the new algorithm in detail and prove a strong theoretical bound on the error of the final hypothesis. We then describe the results of several experiments comparing the new algorithm to a number of other voting methods (including ECOC and pseudoloss-based boosting).

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y$.
Initialize $\tilde{D}_1(i, \ell)$ as in Figure 2.
For $t = 1, \ldots, T$:

- Compute coloring $\mu_t : Y \to \{0, 1\}$.
- Let $U_t = \sum_{i=1}^{m} \sum_{\ell \in Y} \tilde{D}_t(i, \ell) [\![\mu_t(y_i) \neq \mu_t(\ell)]\!]$.
- Let $D_t(i) = \dfrac{\sum_{\ell \in Y} \tilde{D}_t(i, \ell) [\![\mu_t(y_i) \neq \mu_t(\ell)]\!]}{U_t}$.
- Train weak learner on examples $(x_1, \mu_t(y_1)), \ldots, (x_m, \mu_t(y_m))$ weighted according to D_t.
- Get weak hypothesis $h_t : X \to \{0, 1\}$.
- Let $\tilde{h}_t(x) = \{\ell \in Y : h_t(x) = \mu_t(\ell)\}$.
- Let $\tilde{\epsilon}_t$ and α_t be as in Figure 2.
- Compute $\tilde{D}_{t+1}(i, \ell)$ as in Figure 2.

Output the final hypothesis:

$$H_{\text{final}}(x) = \arg\max_{\ell \in Y} \sum_{t=1}^{T} \alpha_t [\![h_t(x) = \mu_t(\ell)]\!].$$

Figure 3: The algorithm AdaBoost.OC combining boosting and output coding.

2 THE NEW ALGORITHM

Both boosting and ECOC work iteratively in rounds by rerunning the weak learning algorithm many times. In boosting, the weak learner is trained on each round on a new distribution or weighting of the training examples. In ECOC, the weak learner is trained on a new partition of the class labels which induces a new binary labeling of the data.

The key idea of the algorithm proposed in this paper is to combine both approaches, i.e., on each round, both to reweight and relabel the data. Generically, then, our algorithm looks like the one in Figure 1.

In the figure, and later in the paper, we use the notation $[\![\pi]\!]$ which we define to be 1 if proposition π holds and 0 otherwise.

The algorithm is given m training examples of the form (x_i, y_i) where x_i is chosen from some space X, and the associated class label y_i is chosen from a set Y of finite cardinality k. On each round t, the algorithm computes a distribution D_t over the training examples, and a function μ_t, which we refer to as a *coloring* and which partitions the label set Y into two parts. The data is then relabeled according to μ_t, and the weak learner trained on this relabeled data weighted according to D_t.

The resulting weak hypothesis is denoted h_t. The goal of the weak learning algorithm is to minimize its training error with respect to the relabeled and reweighted data, that is, to minimize

$$\begin{aligned} \epsilon_t &= \sum_{i=1}^{m} D_t(i) [\![h_t(x_i) \neq \mu_t(y_i)]\!] \\ &= \mathbf{Pr}_{i \sim D_t} \left[h_t(x_i) \neq \mu_t(y_i) \right]. \end{aligned} \quad (1)$$

name	# examples train	# examples test	# classes	# attributes disc.	# attributes cont.	missing values
soybean-small	47	-	4	35	-	-
iris	150	-	3	-	4	-
glass	214	-	7	-	9	-
audiology	226	-	24	69	-	×
soybean-large	307	376	19	35	-	×
vehicle	846	-	4	-	18	-
vowel	528	462	11	-	10	-
segmentation	2310	-	7	-	19	-
splice	3190	-	3	60	-	-
satimage	4435	2000	6	-	36	-
letter	16000	4000	26	-	16	-

Table 1: The benchmark machine learning problems used in the experiments.

Finally, the combined hypothesis H_{final} is computed. This hypothesis can be viewed as a kind of weighted vote of the weak hypotheses. Given an example x, we interpret the binary classification $h_t(x)$ of weak hypothesis h_t as a vote for all of the labels ℓ for which $h_t(x) = \mu_t(\ell)$, i.e., all of the labels with the "color" selected by h_t. This vote is weighted by some real number α_t. The label ℓ receiving the most weighted votes is then chosen as H_{final}'s classification of x. (Ties are broken arbitrarily, and, in our analysis, are counted as errors.)

To complete the description of the algorithm, we need to derive a reasonable choice for the distribution D_t, the coloring μ_t and the coefficients α_t. To do this, we will reduce to the pseudoloss method used by Freund and Schapire [8] in the development of AdaBoost.M2, one of the multiclass versions of their boosting algorithm. This reduction will also lead to an analysis of the resulting algorithm.

2.1 REVIEW OF ADABOOST.M2

We begin with a review of Freund and Schapire's [8] pseudoloss method and of the boosting algorithm AdaBoost.M2, shown in Figure 2. On each round, the boosting algorithm computes a distribution $\tilde{D}_t$ over $\{1, \ldots, m\} \times Y$ such that $\tilde{D}_t(i, y_i) = 0$ for all i. In other words, $\tilde{D}_t$ can be viewed as a distribution over pairs of examples and *in*correct labels. The idea is to enable the boosting algorithm to concentrate the weak learner not only on the hard examples, but also on the incorrect labels which are hardest to distinguish from the correct label.

Given this distribution, the weak learner computes a "soft" hypothesis[1] $\tilde{h}_t : X \to 2^Y$ where 2^Y is the power set of Y. As explained in the introduction, we interpret $\tilde{h}_t(x)$ as a set of "plausible" labels for a given example x. Intuitively, it is easier for the weak learner to identify a set of labels which may plausibly be correct, rather than selecting a single label.

[1] Freund and Schapire [8] allow soft hypotheses to take a more general form as functions mapping $X \times Y$ into $[0, 1]$. The soft hypotheses we consider are equivalent to restricting theirs to have range $\{0, 1\}$. Since this simplifying restriction is a special case of theirs, there is no problem applying their results.

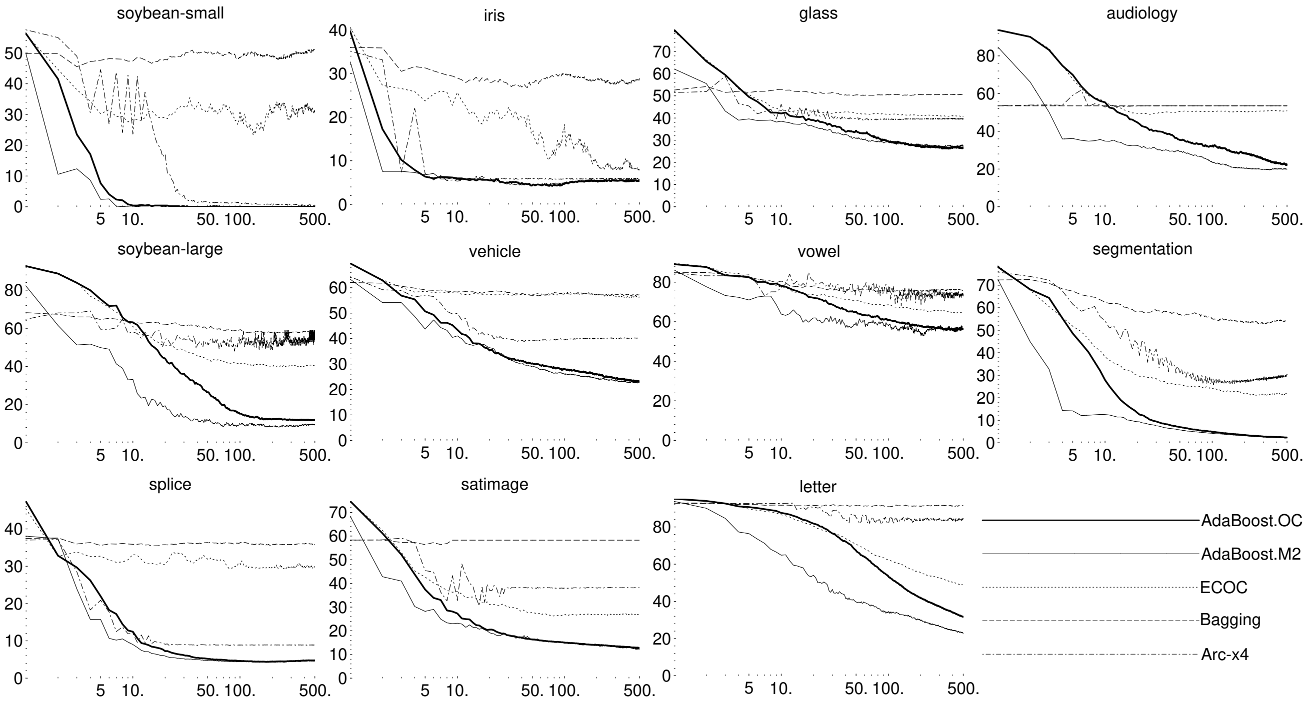

Figure 4: Comparison of several learning methods using FINDATTRTEST as the weak learner.

The goal of the weak learner is to minimize the *pseudoloss*:

$$\tilde{\epsilon}_t = \frac{1}{2}\sum_{i=1}^{m}\sum_{\ell\in Y}\tilde{D}_t(i,\ell)\cdot\left([\![y_i \notin \tilde{h}_t(x_i)]\!] + [\![\ell \in \tilde{h}_t(x_i)]\!]\right).$$

This loss measure penalizes the weak hypothesis for failing to include the correct label y_i in the plausible set associated with example x_i (so that $y_i \notin \tilde{h}_t(x_i)$), and further penalizes each incorrect label $\ell \neq y_i$ which *is* included in the plausible set (so that $\ell \in \tilde{h}_t(x_i)$). (Recall that $\tilde{D}_t(i, y_i) = 0$ so correct labels contribute nothing to the sum.) Note that the pseudoloss is always in $[0, 1]$ and that pseudoloss $1/2$ can be obtained trivially by setting $\tilde{h}_t(x) = \emptyset$ for all x.

Freund and Schapire's [8] ADABOOST.M2 algorithm works by increasing, on each round, the weight placed on examples x_i and incorrect labels ℓ which contribute most to the pseudoloss. The combined hypothesis then chooses the single label which occurs in the largest number of plausible label sets chosen by the weak hypotheses, where the votes of some weak hypotheses count for more than others.

Let $\tilde{\epsilon}_t = 1/2 - \tilde{\gamma}_t$. Freund and Schapire [8, Theorem 11] show that the training error of the combined hypothesis H_{final} of ADABOOST.M2 is bounded by

$$(k-1)\prod_{t=1}^{T}\sqrt{1-4\tilde{\gamma}_t^2} \leq (k-1)\exp\left(-2\sum_{t=1}^{T}\tilde{\gamma}_t^2\right) \quad (2)$$

Thus, if the $\tilde{\gamma}_t$'s are bounded away from $1/2$, then training error goes to zero exponentially fast. Note that, although the weak hypotheses are evaluated with respect to pseudoloss, the final hypothesis H_{final} is analyzed with respect to the usual error measure.

Freund and Schapire also give a method of bounding the generalization error of the combined hypothesis, but, more recently, Schapire et al. [18] have come up with a better analysis of voting methods such as ADABOOST.M2. Their analysis yields bounds on the generalization error in terms of the $\tilde{\gamma}_t$'s, the number of training examples, and a measure of the complexity of the weak hypothesis space (and independent of the number of rounds of boosting).

2.2 OUTPUT CODING AND PSEUDOLOSS

We are now ready to describe the new hybrid algorithm. Returning to Figure 1, suppose that a weak hypothesis $h_t : X \to \{0, 1\}$ has been computed with respect to some coloring $\mu_t : Y \to \{0, 1\}$. As mentioned earlier, the binary classification of h_t on an example x can be viewed as a vote for the labels ℓ for which $h_t(x) = \mu_t(\ell)$, or, said differently, these labels are identified as "plausible" by h_t. Therefore, in reducing to the pseudoloss setting, it is natural to identify h_t with the soft hypothesis $\tilde{h}_t$ defined by:

$$\tilde{h}_t(x) = \mu_t^{-1}(h_t(x)) = \{\ell \in Y : h_t(x) = \mu_t(\ell)\}.$$

Given this choice of soft hypothesis, the update of the distribution $\tilde{D}_t$ is defined for us already by ADABOOST.M2. We will see that D_t and μ_t can in turn be defined sensibly in terms of $\tilde{D}_t$.

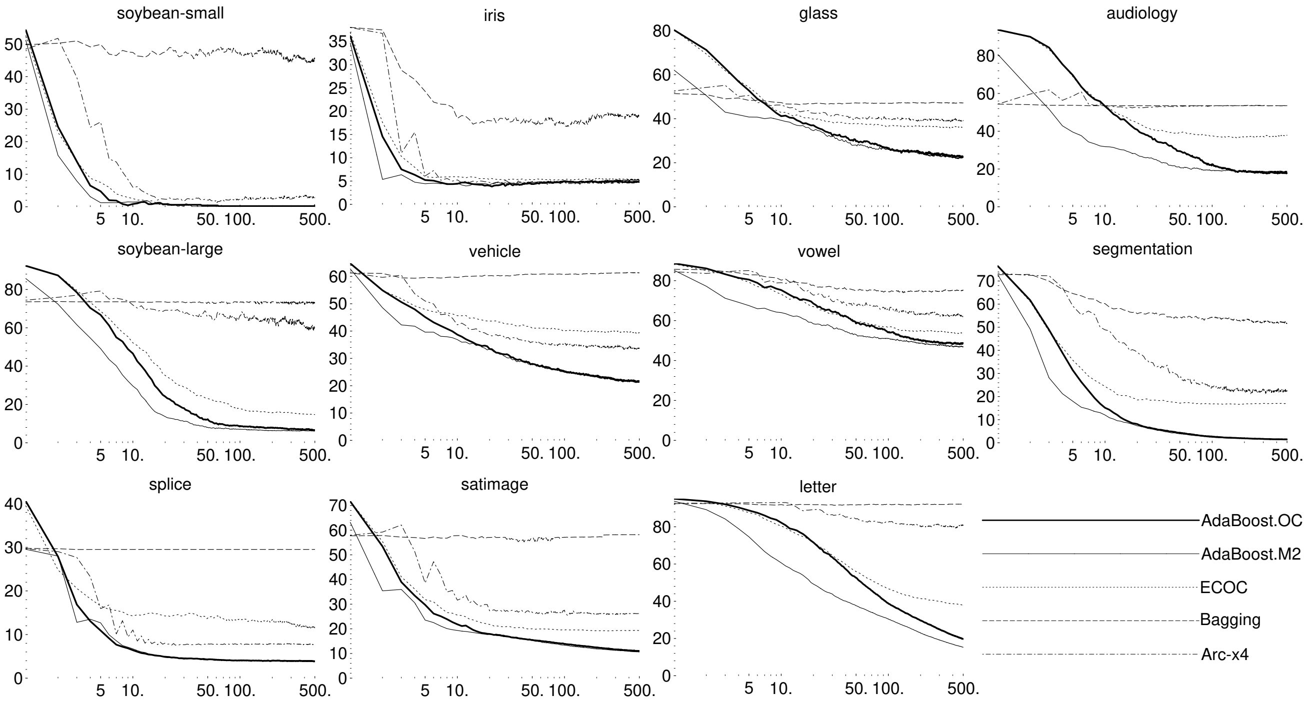

Figure 5: Comparison of several learning methods using FINDDECRULE as the weak learner.

It will be important below to relate the pseudoloss of $\tilde{h}_t$ to the error of h_t. The pseudoloss of $\tilde{h}_t$ can be computed using the following calculation:

$$
\begin{aligned}
&\tfrac{1}{2}\left([\![y_i \notin \tilde{h}_t(x_i)]\!] + [\![\ell \in \tilde{h}_t(x_i)]\!]\right) \\
&\quad = \tfrac{1}{2}\left([\![h_t(x_i) \neq \mu_t(y_i)]\!] + [\![h_t(x_i) = \mu_t(\ell)]\!]\right) \\
&\quad = \begin{cases} \tfrac{1}{2} & \text{if } \mu_t(y_i) = \mu_t(\ell) \\ 1 & \text{if } \mu_t(y_i) \neq h_t(x_i) = \mu_t(\ell) \\ 0 & \text{if } \mu_t(y_i) = h_t(x_i) \neq \mu_t(\ell) \end{cases} \\
&\quad = \tfrac{1}{2}(1 - E_t(i,\ell)) + \eta_t(i) E_t(i,\ell) \qquad (3)
\end{aligned}
$$

where $E_t(i,\ell) = [\![\mu_t(y_i) \neq \mu_t(\ell)]\!]$ indicates if the coloring of ℓ differs from that of the correct label y_i, and $\eta_t(i) = [\![h_t(x_i) \neq \mu_t(y_i)]\!]$ indicates if h_t is incorrect on the ith relabeled example.

A convenient choice for D_t turns out to be the following:

$$D_t(i) = \frac{\sum_{\ell \in Y} \tilde{D}_t(i,\ell) E_t(i,\ell)}{\sum_{i=1}^m \sum_{\ell \in Y} \tilde{D}_t(i,\ell) E_t(i,\ell)}.$$

Let $U_t = \sum_{i=1}^m \sum_{\ell \in Y} \tilde{D}_t(i,\ell) E_t(i,\ell)$, and recall that $\epsilon_t = \sum_{i=1}^m D_t(i)\eta_t(i)$ is the training error of h_t. Then, from Eq. (3), $\tilde{h}_t$'s pseudoloss is

$$
\begin{aligned}
\tilde{\epsilon}_t &= \tfrac{1}{2}\sum_{i=1}^m \sum_{\ell \in Y} \tilde{D}_t(i,\ell)(1 - E_t(i,\ell)) \\
&\quad + \sum_{i=1}^m \eta_t(i) \sum_{\ell \in Y} \tilde{D}_t(i,\ell) E_t(i,\ell) \\
&= \tfrac{1}{2}(1 - U_t) + \epsilon_t U_t \qquad (4)
\end{aligned}
$$

by definition of U_t, ϵ_t and D_t. Thus, with this definition of D_t, the pseudoloss $\tilde{\epsilon}_t$ of $\tilde{h}_t$ can be expressed simply in terms of the error ϵ_t of h_t. Setting $\epsilon_t = 1/2 - \gamma_t$, Eq. (4) becomes

$$\tilde{\epsilon}_t = \tfrac{1}{2} - \gamma_t U_t.$$

So, if h_t has error ϵ_t slightly better than the random guessing error rate of $1/2$, then the pseudoloss of $\tilde{h}_t$ also will be slightly better than $1/2$, provided that $U_t > 0$.

The resulting algorithm, called ADABOOST.OC, is shown in Figure 3. By our method of derivation, this algorithm is in fact a special case of ADABOOST.M2 in which the weak soft hypothesis $\tilde{h}_t$ has a particular form. Therefore, we can immediately apply the results of Freund and Schapire [8] to obtain the following theorem, which is the main theoretical result of this paper:

Theorem 1 *Let $\mu_1, \ldots, \mu_T$ be any sequence of colorings and let $h_1, \ldots, h_T$ be any sequence of weak hypotheses returned by the weak learner. Let $\epsilon_t = 1/2 - \gamma_t$ be the error of h_t with respect to the relabeled and reweighted data on which it was trained (as in Eq. (1)). Let U_t be as in Figure 3. Then the training error of the final hypothesis H_{final} of algorithm* ADABOOST.OC *is bounded by*

$$(k-1)\prod_{t=1}^T \sqrt{1 - 4(\gamma_t U_t)^2} \leq (k-1)\exp\left(-2\sum_{t=1}^T (\gamma_t U_t)^2\right).$$

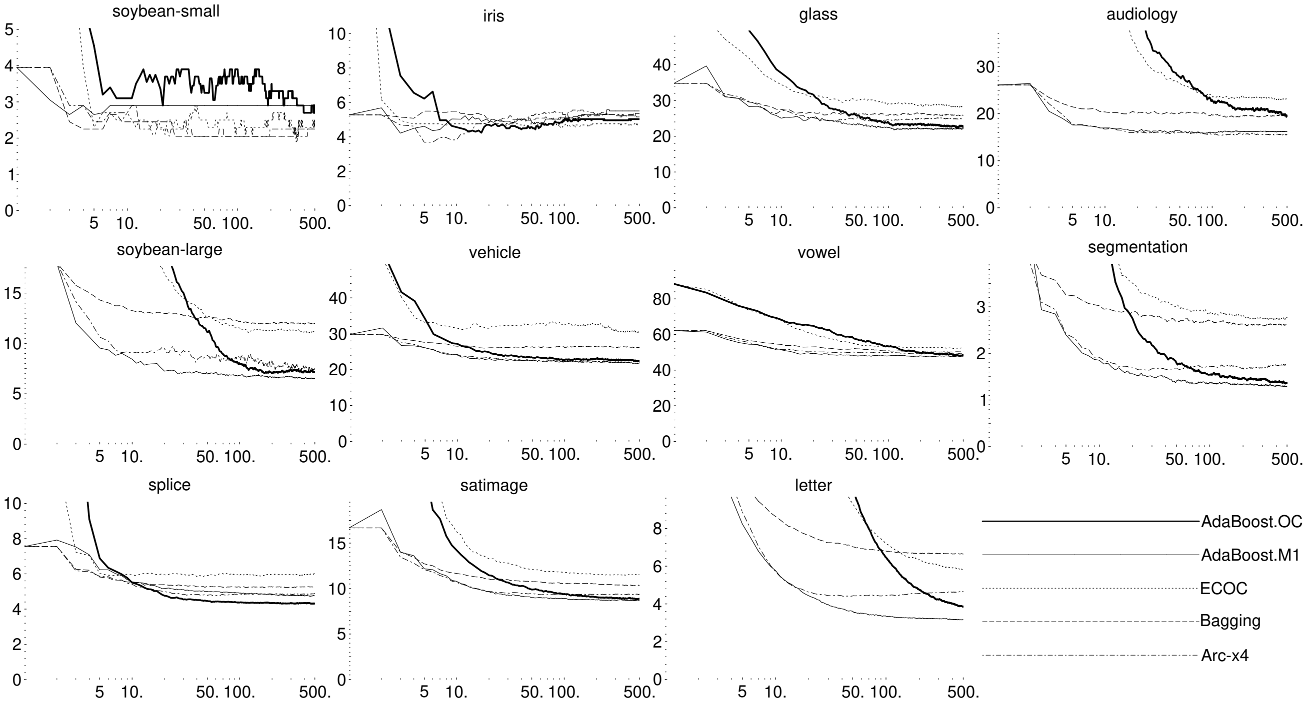

Figure 6: Comparison of several learning methods using C4.5 as the weak learner.

The proof of this theorem follows directly from Eq. (2) and the argument given above on the relationship between the error of the weak hypotheses and the pseudoloss of the associated weak soft hypotheses.

We will show below several methods of choosing a coloring μ_t which gives a value of $U_t \geq 1/2$ (possibly in expectation). Plugging $U_t = 1/2$ into the bound in Theorem 1 gives a bound of

$$(k-1)\prod_{t=1}^{T}\sqrt{1-\gamma_t^2} \leq (k-1)\exp\left(-\tfrac{1}{2}\sum_{t=1}^{T}\gamma_t^2\right)$$

on the training error. Note that this bound approaches zero exponentially fast whenever γ_t is bounded away from zero. Thus, if the weak learner can perform just slightly better than random guessing on the binary problems on which it is trained (so that all the γ_t's are lower bounded by some $\gamma > 0$), then the training error of the final hypothesis can quickly be made arbitrarily small.

Although, for simplicity, we have focused only on the training error, the generalization error can also be bounded using the methods of Schapire et al. [18]. This leads to a bound on the generalization error of the combined hypothesis of the form

$$(k-1)\prod_{t=1}^{T}(1-2\gamma_t U_t)^{1/2-\theta}(1+2\gamma_t U_t)^{1/2+\theta}$$
$$+O\left(\frac{1}{\sqrt{m}}\left(\frac{\log(m/d)(k+d\log(m/d))}{\theta^2}+\log(1/\delta)\right)^{1/2}\right)$$

which holds for all $\theta > 0$ with probability at least $1-\delta$. Here, d is the VC-dimension of the weak hypothesis space used by the weak learner.

The second term will be small when the sample size m is sufficiently large relative to the VC-dimension. And, as with the training error, for small values of θ, the first term drops to zero exponentially fast whenever $\gamma_t U_t$ is bounded away from zero. Details omitted for lack of space.

2.3 CHOOSING THE COLORING

It remains then only to show how to choose a coloring μ_t. From Theorem 1, it is clear that we want to choose μ_t to maximize

$$U_t = \sum_{i=1}^{m}\sum_{\ell\in Y}\tilde{D}_t(i,\ell)[\![\mu_t(y_i) \neq \mu_t(\ell)]\!]. \tag{5}$$

Note that the value of U_t depends only on $\tilde{D}_t$ and μ_t and not on the weak hypothesis. This means that we can attempt to find μ_t maximizing U_t prior to calling the weak learner. Here, we propose a number of options for choosing μ_t.

The simplest option is to choose each value $\mu_t(\ell)$ uniformly and independently at random from $\{0,1\}$ for each label $\ell \in Y$. Then for any $\ell \neq \ell'$, the probability that $\mu_t(\ell) \neq \mu_t(\ell')$ is exactly $1/2$. Therefore, the expected value of U_t is also exactly $1/2$.

A slightly more refined method is to choose μ_t at random but ensuring a (near) even split of the labels. That is, we choose μ_t uniformly at random among all colorings for

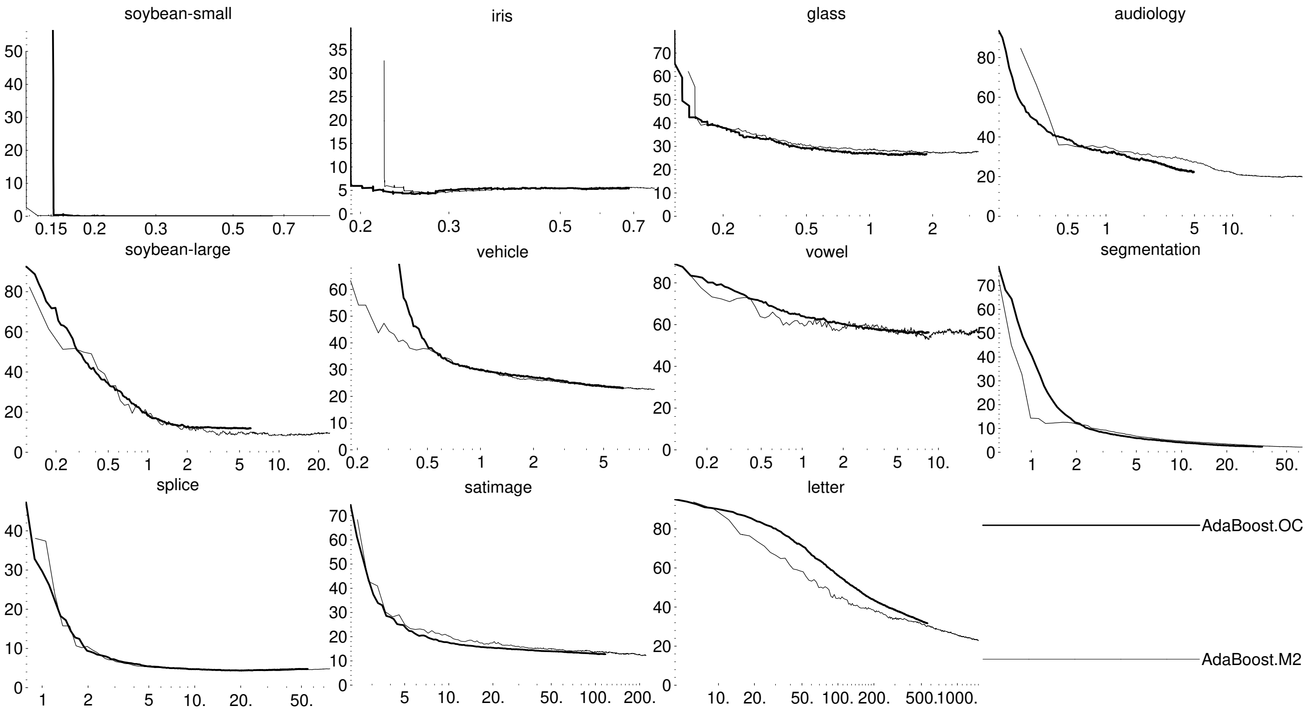

Figure 7: Comparison of computation time versus error rate achieved by ADABOOST.OC and ADABOOST.M2 using FINDATTRTEST as the weak learner.

which exactly $\lfloor k/2 \rfloor$ of the labels are mapped to 0. It can be shown then that, if $\ell \neq \ell'$, then $\mu_t(\ell) \neq \mu_t(\ell')$ with probability $(1/2)(1 + 1/(k-1))$ so the expected value of U_t will also be this value, which is slightly better than $1/2$. This is the method used in the experiments in Section 3.

A last method is to attempt to use combinatorial optimization methods to maximize U_t. Eq. (5) can be rewritten

$$U_t = \sum_{\ell,\ell' \in Y} [\![\mu_t(\ell) \neq \mu_t(\ell')]\!] w_t(\ell, \ell')$$

where $w_t(\ell, \ell') = \sum_{i=1}^{m} \tilde{D}_t(i, \ell)[\![\ell' = y_i]\!]$. Written in this form, it is straightforward to show that maximizing U_t is a special case of the "MAX-CUT" problem, which is known to be NP-complete [14], but for which various, rather sophisticated approximation methods are also known [10, 11]. We did not attempt to use any of these methods, and it is plausible that one of these might improve performance.

In addition, various greedy hill-climbing methods can also be used which guarantee $U_t \geq 1/2$. Experiments using these methods were attempted, but did not give significant improvement over those reported in Section 3 (details not reported).

3 EXPERIMENTS

We tested our method experimentally on a collection of eleven multiclass benchmark problems available from the repository at University of California at Irvine. [2] Some of the characteristics of the benchmarks used are summarized in Table 1. If a test set was already provided, experiments were run 20 times and the results averaged (since many of the learning algorithms used are randomized). If no test set was provided, then 10-fold cross validation was used and rerun 10 times for a total of 100 runs of each algorithm.

For the weak learner, we used three algorithms of varying degrees of expressiveness. The first and simplest, called FINDATTRTEST, outputs a hypothesis that makes its prediction based on the result of a single test comparing one of the attributes to one of its possible values. For discrete attributes, equality is tested; for continuous attributes, a threshold value is compared. The best hypothesis of this form which minimizes error or pseudoloss can be found by a direct and efficient search method. These weak learners are similar in spirit to those studied by Holte [12].

The second weak learner, called FINDDECRULE, outputs a hypothesis which tests on a conjunction of attribute-value comparisons. Such a rule is built up using an entropic potential as in C4.5 and then pruned back using held-out data. This method is based loosely on the rule-formation part of Cohen's RIPPER algorithm [3] and Fürnkranz and Widmer's IREP algorithm [9].

The algorithms FINDATTRTEST and FINDDECRULE are described in more detail by Freund and Schapire [7]. Note

[2] URL: http://www.ics.uci.edu/~mlearn/MLRepository.html

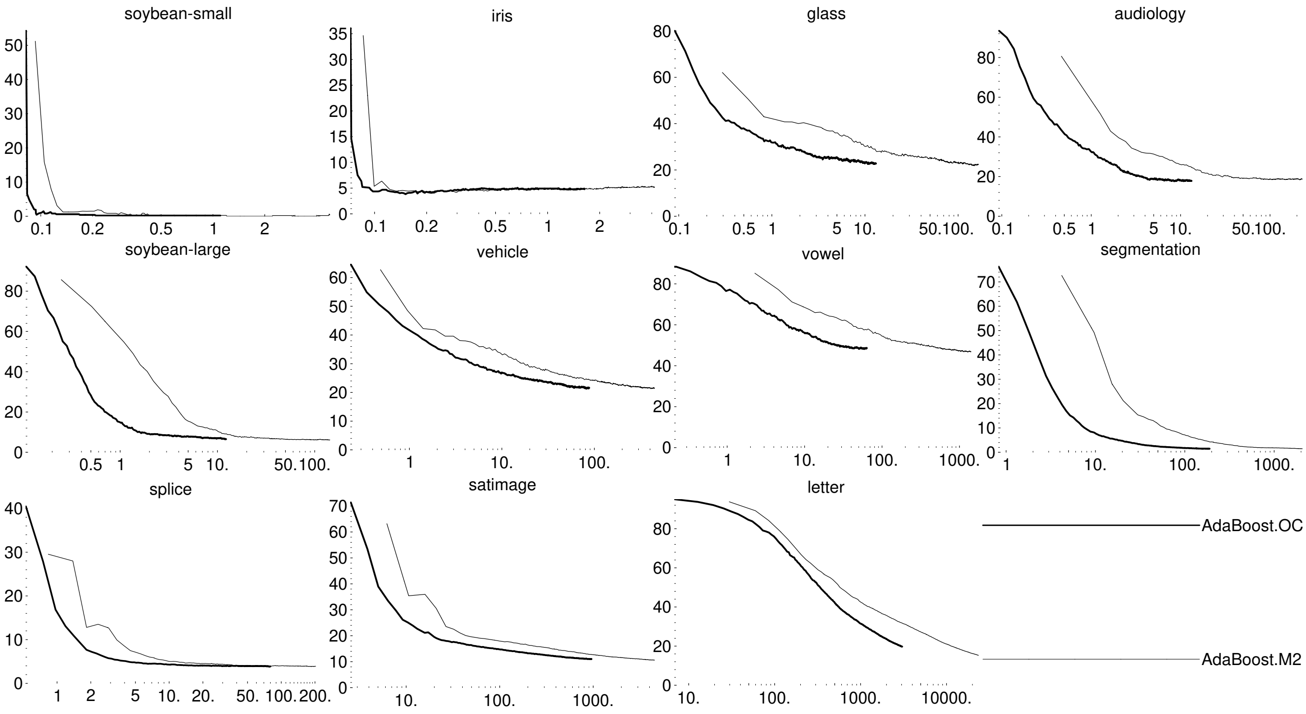

Figure 8: Comparison of computation time versus error rate achieved by ADABOOST.OC and ADABOOST.M2 using FINDDECRULE as the weak learner.

that both algorithms find a single rule for the entire problem, as opposed to learning one rule per class.

The last weak learner tested is Quinlan's C4.5 decision-tree algorithm [16], with all default options and pruning turned on. Also, rather than modify C4.5 to handle weighted examples, on each round t of boosting, we reran C4.5 on (unweighted) examples which were randomly resampled according to D_t.

We first compared ADABOOST.OC to boosting (without ECOC), i.e., to ADABOOST.M2 for FINDATTRTEST and FINDDECRULE, and, for C4.5, to the error-based multiclass version of ADABOOST called ADABOOST.M1. Since our purpose was to derive an algorithm as effective as boosting but one that only requires an error-based (rather than pseudoloss-based) weak learner, we then compared our algorithm to various other methods which combine error-based weak hypotheses. These were:

- Dietterich and Bakiri's ECOC method [4]. However, rather than searching for an error-correcting code, we chose the "output code" at random by selecting each μ_t to be a random (nearly) even split (as described in Section 2.3). Such a random code is highly likely to have error-correcting properties, but it is certainly plausible that a more carefully designed code would perform better than the results reported here.
- Breiman's [1] "bagging" algorithm, which reruns the weak learner on randomly chosen bootstrap samples.
- Breiman's [2] "Arc-x4" algorithm, which, like ADABOOST, adaptively reweights the data, but using a different rule for computing the distribution over examples in a manner that does not require weak hypotheses with error less than $1/2$. This algorithm was shown by Breiman to mimic the performance of ADABOOST when combined with CART, but its theoretical properties are unknown.

Each algorithm was run for 500 rounds. The results are shown in Figures 4, 5 and 6. The x-axis shows the number of rounds, and the y-axis shows the test error of each algorithm (in percent). Note the log scale used in all figures.

For C4.5, there does not seem to be any advantage to using ADABOOST.OC over ADABOOST.M1, and in some cases, such as "audiology," the performance is actually worse for ADABOOST.OC. We also do not expect time savings in this case since we are comparing to ADABOOST.M1 which is already error-based.

For the other, less expressive, weak learners, we see that across all datasets, both ADABOOST.OC and ADABOOST.M2 quickly match the performance of the other three error-based methods, and are usually clearly superior in performance.

Round for round, ADABOOST.OC is often unable to keep up with ADABOOST.M2. However, the savings in computation time can be very significant since error-based algorithms are usually faster than pseudoloss-based algorithms.

Figures 7 and 8 show a plot of computation time (in seconds) versus test error rate for these two algorithms. (The time figures used include both training time and evaluation on test data.) These plots show that, when computation time is limited, the performance of ADABOOST.OC can often surpass that of ADABOOST.M2.

4 CONCLUSION

In conclusion, we have described a new method for training combinations of classifiers that comes with the theoretical guarantees of a boosting algorithm, but, like an output coding algorithm, only requires that the weak learner be capable of handling binary classification problems.

Except for highly expressive weak learners (such as C4.5), the experiments show that this method is a compromise: The test error performance can often, but not always, keep up with pseudoloss-based boosting. On the other hand, the time savings and relative ease of programming the weak learning algorithm can be significant.

ACKNOWLEDGMENTS

Thanks to all those who contributed to the datasets used in this paper.

REFERENCES

[1] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[2] Leo Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California at Berkeley, 1996.

[3] William Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

[4] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, January 1995.

[5] Harris Drucker and Corinna Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems 8*, pages 479–485, 1996.

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

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

[8] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, To appear. An extended abstract appeared in EuroCOLT'95.

[9] Johannes Fürnkranz and Gerhard Widmer. Incremental reduced error pruning. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 70–77, 1994.

[10] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the Association for Computing Machinery*, 42(6):1115–1145, November 1995.

[11] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. In *37th Annual Symposium on Foundations of Computer Science*, pages 339–348, 1996.

[12] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–91, 1993.

[13] Jeffrey C. Jackson and Mark W. Craven. Learning sparse perceptrons. In *Advances in Neural Information Processing Systems 8*, pages 654–660, 1996.

[14] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[15] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

[16] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

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

[18] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Machine Learning: Proceedings of the Fourteenth International Conference*, 1997.