Machine Learning, 37(3):297-336, 1999.

Improved Boosting Algorithms Using Confidence-rated Predictions

ROBERT E. SCHAPIRE schapire@research.att.com
AT&T Labs, Shannon Laboratory, 180 Park Avenue, Room A279, Florham Park, NJ 07932-0971

YORAM SINGER singer@research.att.com
AT&T Labs, Shannon Laboratory, 180 Park Avenue, Room A277, Florham Park, NJ 07932-0971

Abstract. We describe several improvements to Freund and Schapire's AdaBoost boosting algorithm, particularly in a setting in which hypotheses may assign confidences to each of their predictions. We give a simplified analysis of AdaBoost in this setting, and we show how this analysis can be used to find improved parameter settings as well as a refined criterion for training weak hypotheses. We give a specific method for assigning confidences to the predictions of decision trees, a method closely related to one used by Quinlan. This method also suggests a technique for growing decision trees which turns out to be identical to one proposed by Kearns and Mansour.

We focus next on how to apply the new boosting algorithms to multiclass classification problems, particularly to the multi-label case in which each example may belong to more than one class. We give two boosting methods for this problem, plus a third method based on output coding. One of these leads to a new method for handling the single-label case which is simpler but as effective as techniques suggested by Freund and Schapire. Finally, we give some experimental results comparing a few of the algorithms discussed in this paper.

Keywords: Boosting algorithms, multiclass classification, output coding, decision trees

1. Introduction

Boosting is a method of finding a highly accurate hypothesis (classification rule) by combining many "weak" hypotheses, each of which is only moderately accurate. Typically, each weak hypothesis is a simple rule which can be used to generate a predicted classification for any instance. In this paper, we study boosting in an extended framework in which each weak hypothesis generates not only predicted classifications, but also self-rated confidence scores which estimate the reliability of each of its predictions.

There are two essential questions which arise in studying this problem in the boosting paradigm. First, how do we modify known boosting algorithms designed to handle only simple predictions to use confidence-rated predictions in the most effective manner possible? Second, how should we design weak learners whose predictions are confidence-rated in the manner described above? In this paper, we give answers to both of these questions. The result is a powerful set of boosting methods for handling more expressive weak hypotheses, as well as an advanced methodology for designing weak learners appropriate for use with boosting algorithms.

We base our work on Freund and Schapire's (1997) AdaBoost algorithm which has received extensive empirical and theoretical study (Bauer & Kohavi, to appear; Breiman, 1998; Dietterich, to appear; Dietterich & Bakiri, 1995; Drucker & Cortes, 1996; Freund & Schapire, 1996; Maclin & Opitz, 1997; Margineantu & Dietterich, 1997; Quinlan, 1996; Schapire, 1997; Schapire, Freund, Bartlett, & Lee, 1998; Schwenk & Bengio, 1998).

To boost using confidence-rated predictions, we propose a generalization of AdaBoost in which the main parameters α_t are tuned using one of a number of methods that we describe in detail. Intuitively, the α_t's control the influence of each of the weak hypotheses. To determine the proper tuning of these parameters, we begin by presenting a streamlined version of Freund and Schapire's analysis which provides a clean upper bound on the training error of AdaBoost when the parameters α_t are left unspecified. For the purposes of minimizing training error, this analysis provides an immediate clarification of the criterion that should be used in setting α_t. As discussed below, this analysis also provides the criterion that should be used by the weak learner in formulating its weak hypotheses.

Based on this analysis, we give a number of methods for choosing α_t. We show that the optimal tuning (with respect to our criterion) of α_t can be found numerically in general, and we give exact methods of setting α_t in special cases.

Freund and Schapire also considered the case in which the individual predictions of the weak hypotheses are allowed to carry a confidence. However, we show that their setting of α_t is only an approximation of the optimal tuning which can be found using our techniques.

We next discuss methods for designing weak learners with confidence-rated predictions using the criterion provided by our analysis. For weak hypotheses which partition the instance space into a small number of equivalent prediction regions, such as decision trees, we present and analyze a simple method for automatically assigning a level of confidence to the predictions which are made within each region. This method turns out to be closely related to a heuristic method proposed by Quinlan (1996) for boosting decision trees. Our analysis can be viewed as a partial theoretical justification for his experimentally successful method.

Our technique also leads to a modified criterion for selecting such domain-partitioning weak hypotheses. In other words, rather than the weak learner simply choosing a weak hypothesis with low training error as has usually been done in the past, we show that, theoretically, our methods work best when combined with a weak learner which minimizes an alternative measure of "badness." For growing decision trees, this measure turns out to be identical to one earlier proposed by Kearns and Mansour (1996).

Although we primarily focus on minimizing training error, we also outline methods that can be used to analyze generalization error as well.

Next, we show how to extend the methods described above for binary classification problems to the multiclass case, and, more generally, to the *multi-label* case in which each example may belong to more than one class. Such problems arise naturally, for instance, in text categorization problems where the same document (say, a news article) may easily be relevant to more than one topic (such as politics, sports, etc.).

Freund and Schapire (1997) gave two algorithms for boosting multiclass problems, but neither was designed to handle the multi-label case. In this paper, we present two new extensions of AdaBoost for multi-label problems. In both cases, we show how to apply the results presented in the first half of the paper to these new extensions.

In the first extension, the learned hypothesis is evaluated in terms of its ability to predict a good approximation of the set of labels associated with a given instance. As a special case, we obtain a novel boosting algorithm for multiclass problems in the more conventional single-label case. This algorithm is simpler but apparently as effective as the methods given by Freund and Schapire. In addition, we propose and analyze a modification of this method which combines these techniques with Dietterich and Bakiri's (1995) output-coding method. (Another method of combining boosting and output coding was proposed by Schapire (1997). Although superficially similar, his method is in fact quite different from what is presented here.)

In the second extension to multi-label problems, the learned hypothesis instead predicts, for a given instance, a ranking of the labels, and it is evaluated based on its ability to place the correct labels high in this ranking. Freund and Schapire's AdaBoost.M2 is a special case of this method for single-label problems.

Although the primary focus of this paper is on theoretical issues, we give some experimental results comparing a few of the new algorithms. We obtain especially dramatic improvements in performance when a fairly large amount of data is available, such as large text categorization problems.

2. A Generalized Analysis of Adaboost

Let $S = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ be a sequence of training examples where each *instance* x_i belongs to a *domain* or *instance space* $\mathcal{X}$, and each *label* y_i belongs to a finite *label space* $\mathcal{Y}$. For now, we focus on binary classification problems in which $\mathcal{Y} = \{-1, +1\}$.

We assume access to a *weak* or *base* learning algorithm which accepts as input a sequence of training examples S along with a distribution D over $\{1, \ldots, m\}$, i.e., over the indices of S. Given such input, the weak learner computes a *weak* (or *base*) *hypothesis* h. In general, h has the form $h : \mathcal{X} \to \mathbb{R}$. We interpret the sign of $h(x)$ as the predicted label (-1 or $+1$) to be assigned to instance x, and the magnitude $|h(x)|$ as the "confidence" in this prediction. Thus, if $h(x)$ is close to or far from zero, it is interpreted as a low or high confidence prediction. Although the range of h may generally include all real numbers, we will sometimes restrict this range.

The idea of boosting is to use the weak learner to form a highly accurate prediction rule by calling the weak learner repeatedly on different distributions over the training examples. A slightly generalized version of Freund and Schapire's AdaBoost algorithm is shown in Figure 1. The main effect of AdaBoost's update rule, assuming $\alpha_t > 0$, is to decrease or increase the weight of training examples classified correctly or incorrectly by h_t (i.e., examples i for which y_i and $h_t(x_i)$ agree or disagree in sign).

Our version differs from Freund and Schapire's in that (1) weak hypotheses can have range over all of $\mathbb{R}$ rather than the restricted range $[-1, +1]$ assumed by Freund and Schapire; and (2) whereas Freund and Schapire prescribe a specific choice of α_t, we leave this choice unspecified and discuss various tunings below. Despite these differences, we continue to refer to the algorithm of Figure 1 as "AdaBoost."

As discussed below, when the range of each h_t is restricted to $[-1, +1]$, we can choose α_t appropriately to obtain Freund and Schapire's original AdaBoost algorithm (ignoring superficial differences in notation). Here, we give a simplified analysis of the algorithm in which α_t is left unspecified. This analysis yields an improved and more general method for choosing α_t.

Let

Given: $(x_1, y_1), \ldots, (x_m, y_m)$; $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Figure 1. A generalized version of AdaBoost.

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

so that $H(x) = \operatorname{sign}(f(x))$. Also, for any predicate π, let $[\![\pi]\!]$ be 1 if π holds and 0 otherwise. We can prove the following bound on the training error of H.

THEOREM 1 *Assuming the notation of Figure 1, the following bound holds on the training error of H:*

$$\frac{1}{m} \left|\{i : H(x_i) \neq y_i\}\right| \leq \prod_{t=1}^{T} Z_t.$$

Proof: By unraveling the update rule, we have that

$$\begin{aligned} D_{T+1}(i) &= \frac{\exp\left(-\sum_t \alpha_t y_i h_t(x_i)\right)}{m \prod_t Z_t} \\ &= \frac{\exp\left(-y_i f(x_i)\right)}{m \prod_t Z_t}. \end{aligned} \tag{1}$$

Moreover, if $H(x_i) \neq y_i$ then $y_i f(x_i) \leq 0$ implying that $\exp(-y_i f(x_i)) \geq 1$. Thus,

$$[\![H(x_i) \neq y_i]\!] \leq \exp(-y_i f(x_i)). \tag{2}$$

Combining Eqs. (1) and (2) gives the stated bound on training error since

$$\begin{aligned}\frac{1}{m}\sum_i [\![H(x_i) \neq y_i]\!] &\leq \frac{1}{m}\sum_i \exp(-y_i f(x_i)) \\ &= \sum_i \left(\prod_t Z_t\right) D_{T+1}(i) \\ &= \prod_t Z_t \ .\end{aligned}$$

■

The important consequence of Theorem 1 is that, in order to minimize training error, a reasonable approach might be to greedily minimize the bound given in the theorem by minimizing Z_t on each round of boosting. We can apply this idea both in the choice of α_t and as a general criterion for the choice of weak hypothesis h_t.

Before proceeding with a discussion of how to apply this principle, however, we digress momentarily to give a slightly different view of AdaBoost. Let $\mathcal{H} = \{g_1, \ldots, g_N\}$ be the space of all possible weak hypotheses, which, for simplicity, we assume for the moment to be finite. Then AdaBoost attempts to find a linear threshold of these weak hypotheses which gives good predictions, i.e., a function of the form

$$H(x) = \text{sign}\left(\sum_{j=1}^{N} a_j g_j(x)\right) .$$

By the same argument used in Theorem 1, it can be seen that the number of training mistakes of H is at most

$$\sum_{i=1}^{m} \exp\left(-y_i \sum_{j=1}^{N} a_j g_j(x_i)\right) . \tag{3}$$

AdaBoost can be viewed as a method for minimizing the expression in Eq. (3) over the coefficients a_j by a greedy coordinate-wise search: On each round t, a coordinate j is chosen corresponding to h_t, that is, $h_t = g_j$. Next, the value of the coefficient a_j is modified by adding α_t to it; all other coefficient are left unchanged. It can be verified that the quantity Z_t measures exactly the ratio of the new to the old value of the exponential sum in Eq. (3) so that $\prod_t Z_t$ is the final value of this expression (assuming we start with all a_j's set to zero).

See Friedman, Hastie and Tibshirani (1998) for further discussion of the rationale for minimizing Eq. (3), including a connection to logistic regression. See also Appendix A for further comments on how to minimize expressions of this form.

3. Choosing α_t

To simplify notation, let us fix t and let $u_i = y_i h_t(x_i)$, $Z = Z_t$, $D = D_t$, $h = h_t$ and $\alpha = \alpha_t$. In the following discussion, we assume without loss of generality that $D(i) \neq 0$ for all i. Our goal is to find α which minimizes or approximately minimizes Z as a function of α. We describe a number of methods for this purpose.

3.1. *Deriving Freund and Schapire's choice of* $\boldsymbol{\alpha_t}$

We begin by showing how Freund and Schapire's (1997) version of AdaBoost can be derived as a special case of our new version. For weak hypotheses h with range $[-1,+1]$, their choice of α can be obtained by approximating Z as follows:

$$\begin{aligned} Z &= \sum_i D(i) e^{-\alpha u_i} \\ &\leq \sum_i D(i) \left(\frac{1+u_i}{2} e^{-\alpha} + \frac{1-u_i}{2} e^{\alpha} \right) . \end{aligned} \tag{4}$$

This upper bound is valid since $u_i \in [-1,+1]$, and is in fact exact if h has range $\{-1,+1\}$ (so that $u_i \in \{-1,+1\}$). (A proof of the bound follows immediately from the convexity of $e^{-\alpha x}$ for any constant $\alpha \in \mathbb{R}$.) Next, we can analytically choose α to minimize the right hand side of Eq. (4) giving

$$\alpha = \tfrac{1}{2} \ln \left(\frac{1+r}{1-r} \right)$$

where $r = \sum_i D(i) u_i$. Plugging into Eq. (4), this choice gives the upper bound

$$Z \leq \sqrt{1-r^2}.$$

We have thus proved the following corollary of Theorem 1 which is equivalent to Freund and Schapire's (1997) Theorem 6:

COROLLARY 1 ((FREUND & SCHAPIRE, 1997)) *Using the notation of Figure 1, assume each* h_t *has range* $[-1,+1]$ *and that we choose*

$$\alpha_t = \tfrac{1}{2} \ln \left(\frac{1+r_t}{1-r_t} \right)$$

where

$$r_t = \sum_i D_t(i) y_i h_t(x_i) = \mathrm{E}_{i \sim D_t} \left[y_i h_t(x_i) \right].$$

Then the training error of H *is at most*

$$\prod_{t=1}^{T} \sqrt{1-r_t^2}.$$

Thus, with this setting of α_t, it is reasonable to try to find h_t that maximizes $|r_t|$ on each round of boosting. This quantity r_t is a natural measure of the correlation of the predictions of h_t and the labels y_i with respect to the distribution D_t. It is closely related to ordinary error since, if h_t has range $\{-1,+1\}$ then

$$\Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right] = \frac{1 - r_t}{2}$$

so maximizing r_t is equivalent to minimizing error. More generally, if h_t has range $[-1, +1]$ then $(1-r_t)/2$ is equivalent to the definition of error used by Freund and Schapire (ϵ_t in their notation).

The approximation used in Eq. (4) is essentially a linear upper bound of the function $e^{-\alpha x}$ on the range $x \in [-1, +1]$. Clearly, other upper bounds which give a tighter approximation could be used instead, such as a quadratic or piecewise-linear approximation.

3.2. *A numerical method for the general case*

We next give a general numerical method for exactly minimizing Z with respect to α. Recall that our goal is to find α which minimizes

$$Z(\alpha) = Z = \sum_i D(i) e^{-\alpha u_i}.$$

The first derivative of Z is

$$\begin{aligned} Z'(\alpha) = \frac{dZ}{d\alpha} &= -\sum_i D(i) u_i e^{-\alpha u_i} \\ &= -Z \sum_i D_{t+1}(i) u_i \end{aligned}$$

by definition of D_{t+1}. Thus, if D_{t+1} is formed using the value of α_t which minimizes Z_t (so that $Z'(\alpha) = 0$), then we will have that

$$\sum_i D_{t+1}(i) u_i = \mathrm{E}_{i \sim D_{t+1}}\left[y_i h_t(x_i)\right] = 0.$$

In words, this means that, with respect to distribution D_{t+1}, the weak hypothesis h_t will be exactly uncorrelated with the labels y_i.

It can easily be verified that $Z''(\alpha) = d^2 Z / d\alpha^2$ is strictly positive for all $\alpha \in \mathbb{R}$ (ignoring the trivial case that $u_i = 0$ for all i). Therefore, $Z'(\alpha)$ can have at most one zero. (See also Appendix A.)

Moreover, if there exists i such that $u_i < 0$ then $Z'(\alpha) \to \infty$ as $\alpha \to \infty$. Similarly, $Z'(\alpha) \to -\infty$ as $\alpha \to -\infty$ if $u_i > 0$ for some i. This means that $Z'(\alpha)$ has at least one root, except in the degenerate case that all non-zero u_i's are of the same sign. Furthermore, because $Z'(\alpha)$ is strictly increasing, we can numerically find the unique minimum of $Z(\alpha)$ by a simple binary search, or more sophisticated numerical methods.

Summarizing, we have argued the following:

THEOREM 2

1. *Assume the set $\{y_i h_t(x_i) : i = 1, \ldots, m\}$ includes both positive and negative values. Then there exists a unique choice of α_t which minimizes Z_t.*

2. *For this choice of α_t, we have that*

$$\mathrm{E}_{i \sim D_{t+1}}\left[y_i h_t(x_i)\right] = 0 .$$

3.3. An analytic method for weak hypotheses that abstain

We next consider a natural special case in which the choice of α_t can be computed analytically rather than numerically.

Suppose that the range of each weak hypothesis h_t is now restricted to $\{-1, 0, +1\}$. In other words, a weak hypothesis can make a definitive prediction that the label is -1 or $+1$, or it can "abstain" by predicting 0. No other levels of confidence are allowed. By allowing the weak hypothesis to effectively say "I don't know," we introduce a model analogous to the "specialist" model of Blum (1997), studied further by Freund et al. (1997).

For fixed t, let W_0, W_{-1}, W_{+1} be defined by

$$W_b = \sum_{i: u_i = b} D(i)$$

for $b \in \{-1, 0, +1\}$, where, as before, $u_i = y_i h_t(x_i)$, and where we continue to omit the subscript t when clear from context. Also, for readability of notation, we will often abbreviate subscripts $+1$ and -1 by the symbols $+$ and $-$ so that W_{+1} is written W_+, and W_{-1} is written W_-. We can calculate Z as:

$$\begin{aligned} Z &= \sum_i D(i) e^{-\alpha u} \\ &= \sum_{b \in \{-1,0,+1\}} \sum_{i: u_i = b} D(i) e^{-\alpha b} \\ &= W_0 + W_- e^{\alpha} + W_+ e^{-\alpha}. \end{aligned}$$

It can easily be verified that Z is minimized when

$$\alpha = \tfrac{1}{2} \ln \left(\frac{W_+}{W_-} \right).$$

For this setting of α, we have

$$Z = W_0 + 2\sqrt{W_- W_+}. \tag{5}$$

For this case, Freund and Schapire's original AdaBoost algorithm would instead have made the more conservative choice

$$\alpha = \tfrac{1}{2} \ln \left(\frac{W_+ + \frac{1}{2} W_0}{W_- + \frac{1}{2} W_0} \right)$$

giving a value of Z which is necessarily inferior to Eq. (5), but which Freund and Schapire (1997) are able to upper bound by

$$Z \le 2\sqrt{(W_- + \tfrac{1}{2} W_0)(W_+ + \tfrac{1}{2} W_0)}. \tag{6}$$

If $W_0 = 0$ (so that h has range $\{-1, +1\}$), then the choices of α and resulting values of Z are identical.

4. A Criterion for Finding Weak Hypotheses

So far, we have only discussed using Theorem 1 to choose α_t. In general, however, this theorem can be applied more broadly to guide us in the design of weak learning algorithms which can be combined more powerfully with boosting.

In the past, it has been assumed that the goal of the weak learning algorithm should be to find a weak hypothesis h_t with a small number of errors with respect to the given distribution D_t over training samples. The results above suggest, however, that a different criterion can be used. In particular, we can attempt to greedily minimize the upper bound on training error given in Theorem 1 by minimizing Z_t on each round. Thus, the weak learner should attempt to find a weak hypothesis h_t which minimizes

$$Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i)).$$

This expression can be simplified by folding α_t into h_t, in other words, by assuming without loss of generality that the weak learner can freely scale any weak hypothesis h by any constant factor $\alpha \in \mathbb{R}$. Then (omitting t subscripts), the weak learner's goal now is to minimize

$$Z = \sum_i D(i) \exp(-y_i h(x_i)). \tag{7}$$

For some algorithms, it may be possible to make appropriate modifications to handle such a "loss" function directly. For instance, gradient-based algorithms, such as backprop, can easily be modified to minimize Eq. (7) rather than the more traditional mean squared error.

We show how decision-tree algorithms can be modified based on the new criterion for finding good weak hypotheses.

4.1. Domain-partitioning weak hypotheses

We focus now on weak hypotheses which make their predictions based on a partitioning of the domain $\mathcal{X}$. To be more specific, each such weak hypothesis is associated with a partition of $\mathcal{X}$ into disjoint blocks $X_1, \ldots, X_N$ which cover all of $\mathcal{X}$ and for which $h(x) = h(x')$ for all $x, x' \in X_j$. In other words, h's prediction depends only on which block X_j a given instance falls into. A prime example of such a hypothesis is a decision tree whose leaves define a partition of the domain.

Suppose that $D = D_t$ and that we have already found a partition $X_1, \ldots, X_N$ of the space. What predictions should be made for each block of the partition? In other words, how do we find a function $h : \mathcal{X} \to \mathbb{R}$ which respects the given partition and which minimizes Eq. (7)?

Let $c_j = h(x)$ for $x \in X_j$. Our goal is to find appropriate choices for c_j. For each j and for $b \in \{-1, +1\}$, let

$$W_b^j = \sum_{i: x_i \in X_j \wedge y_i = b} D(i) = \Pr_{i \sim D}\left[x_i \in X_j \wedge y_i = b\right]$$

be the weighted fraction of examples which fall in block j with label b. Then Eq. (7) can be rewritten

$$\begin{aligned} Z &= \sum_j \sum_{i:x_i \in X_j} D(i) \exp(-y_i c_j) \\ &= \sum_j \left(W_+^j e^{-c_j} + W_-^j e^{c_j} \right). \end{aligned} \tag{8}$$

Using standard calculus, we see that this is minimized when

$$c_j = \tfrac{1}{2} \ln \left(\frac{W_+^j}{W_-^j} \right). \tag{9}$$

Plugging into Eq. (8), this choice gives

$$Z = 2 \sum_j \sqrt{W_+^j W_-^j}. \tag{10}$$

Note that the sign of c_j is equal to the (weighted) majority class within block j. Moreover, c_j will be close to zero (a low confidence prediction) if there is a roughly equal split of positive and negative examples in block j. Likewise, c_j will be far from zero if one label strongly predominates.

A similar scheme was previously proposed by Quinlan (1996) for assigning confidences to the predictions made at the leaves of a decision tree. Although his scheme differed in the details, we feel that our new theory provides some partial justification for his method.

The criterion given by Eq. (10) can also be used as a splitting criterion in growing a decision tree, rather than the Gini index or an entropic function. In other words, the decision tree could be built by greedily choosing the split which causes the greatest drop in the value of the function given in Eq. (10). In fact, exactly this splitting criterion was proposed by Kearns and Mansour (1996). Furthermore, if one wants to boost more than one decision tree then each tree can be built using the splitting criterion given by Eq. (10) while the predictions at the leaves of the boosted trees are given by Eq. (9).

4.2. Smoothing the predictions

The scheme presented above requires that we predict as in Eq. (9) on block j. It may well happen that W_-^j or W_+^j is very small or even zero, in which case c_j will be very large or infinite in magnitude. In practice, such large predictions may cause numerical problems. In addition, there may be theoretical reasons to suspect that large, overly confident predictions will increase the tendency to overfit.

To limit the magnitudes of the predictions, we suggest using instead the "smoothed" values

$$c_j = \tfrac{1}{2} \ln \left(\frac{W_+^j + \varepsilon}{W_-^j + \varepsilon} \right)$$

for some appropriately small positive value of ε. Because W_-^j and W_+^j are both bounded between 0 and 1, this has the effect of bounding $|c_j|$ by

$$\tfrac{1}{2}\ln\left(\frac{1+\varepsilon}{\varepsilon}\right) \approx \tfrac{1}{2}\ln(1/\varepsilon).$$

Moreover, this smoothing only slightly weakens the value of Z since, plugging into Eq. (8) gives

$$\begin{aligned}
Z &= \sum_j \left(W_+^j \sqrt{\frac{W_-^j + \varepsilon}{W_+^j + \varepsilon}} + W_-^j \sqrt{\frac{W_+^j + \varepsilon}{W_-^j + \varepsilon}} \right) \\
&\leq \sum_j \left(\sqrt{(W_-^j + \varepsilon)W_+^j} + \sqrt{(W_+^j + \varepsilon)W_-^j} \right) \\
&\leq \sum_j \left(2\sqrt{W_-^j W_+^j} + \sqrt{\varepsilon W_+^j} + \sqrt{\varepsilon W_-^j} \right) \\
&\leq 2\sum_j \sqrt{W_-^j W_+^j} + \sqrt{2N\varepsilon}\,. \qquad (11)
\end{aligned}$$

In the second inequality, we used the inequality $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$ for nonnegative x and y. In the last inequality, we used the fact that

$$\sum_j (W_-^j + W_+^j) = 1,$$

which implies

$$\sum_j \left(\sqrt{W_-^j} + \sqrt{W_+^j} \right) \leq \sqrt{2N}\,.$$

(Recall that N is the number of blocks in the partition.) Thus, comparing Eqs. (11) and (10), we see that Z will not be greatly degraded by smoothing if we choose $\varepsilon \ll 1/(2N)$. In our experiments, we have typically used ε on the order of $1/m$ where m is the number of training examples.

5. Generalization Error

So far, we have only focused on the training error, even though our primary objective is to achieve low generalization error.

Two methods of analyzing the generalization error of AdaBoost have been proposed. The first, given by Freund and Schapire (1997), uses standard VC-theory to bound the generalization error of the final hypothesis in terms of its training error and an additional term which is a function of the VC-dimension of the final hypothesis class and the number of training examples. The VC-dimension of the final hypothesis class can be computed using the methods of Baum and Haussler (1989). Interpretting the derived upper bound as a qualitative prediction of behavior, this analysis suggests that AdaBoost is more likely to overfit if run for too many rounds.

Schapire et al. (1998) proposed an alternative analysis to explain AdaBoost's empirically observed resistance to overfitting. Following the work of Bartlett (1998), this method is based on the "margins" achieved by the final hypothesis on the training examples. The margin is a measure of the "confidence" of the prediction. Schapire et al. show that larger margins imply lower generalization error — regardless of the number of rounds. Moreover, they show that AdaBoost tends to increase the margins of the training examples.

To a large extent, their analysis can be carried over to the current context, which is the focus of this section. As a first step in applying their theory, we assume that each weak hypothesis h_t has bounded range. Recall that the final hypothesis has the form

$$H(x) = \mathrm{sign}(f(x))$$

where

$$f(x) = \sum_t \alpha_t h_t(x).$$

Since the h_t's are bounded and since we only care about the sign of f, we can rescale the h_t's and normalize the α_t's allowing us to assume without loss of generality that each $h_t : \mathcal{X} \to [-1,+1]$, each $\alpha_t \in [0,1]$ and $\sum_t \alpha_t = 1$. Let us also assume that each h_t belongs to a hypothesis space $\mathcal{H}$.

Schapire et al. define the *margin* of a labeled example (x, y) to be $yf(x)$. The margin then is in $[-1,+1]$, and is positive if and only if H makes a correct prediction on this example. We further regard the magnitude of the margin as a measure of the confidence of H's prediction.

Schapire et al.'s results can be applied directly in the present context only in the special case that each $h \in \mathcal{H}$ has range $\{-1,+1\}$. This case is not of much interest, however, since our focus is on weak hypotheses with real-valued predictions. To extend the margins theory, then, let us define d to be the *pseudodimension* of $\mathcal{H}$ (for definitions, see, for instance, Haussler (1992)). Then using the method sketched in Section 2.4 of Schapire et al. together with Haussler and Long's (1995) Lemma 13, we can prove the following upper bound on generalization error which holds with probability $1-\delta$ for all $\theta > 0$ and for all f of the form above:

$$\mathrm{Pr}_S\left[yf(x) \le \theta\right] + O\left(\frac{1}{\sqrt{m}}\left(\frac{d\log^2(m/d)}{\theta^2} + \log(1/\delta)\right)^{1/2}\right).$$

Here, Pr_S denotes probability with respect to choosing an example (x, y) uniformly at random from the training set. Thus, the first term is the fraction of training examples with margin at most θ. A proof outline of this bound was communicated to us by Peter Bartlett and is provided in Appendix B.

Note that, as mentioned in Section 4.2, this margin-based analysis suggests that it may be a bad idea to allow weak hypotheses which sometimes make predictions that are very large in magnitude. If $|h_t(x)|$ is very large for some x, then rescaling h_t leads to a very large coefficient α_t which, in turn, may overwhelm the other coefficients and so may dramatically reduce the margins of some of the training examples. This, in turn, according to our theory, can have a detrimental effect on the generalization error.

It remains to be seen if this theoretical effect will be observed in practice, or, alternatively, if an improved theory can be developed.

6. Multiclass, Multi-label Classification Problems

We next show how some of these methods can be extended to the multiclass case in which there may be more than two possible labels or classes. Moreover, we will consider the more general *multi-label* case in which a single example may belong to any number of classes.

Formally, we let $\mathcal{Y}$ be a finite set of labels or classes, and let $k = |\mathcal{Y}|$. In the traditional classification setting, each example $x \in \mathcal{X}$ is assigned a single class $y \in \mathcal{Y}$ (possibly via a stochastic process) so that labeled examples are pairs (x, y). The goal then, typically, is to find a hypothesis $H : \mathcal{X} \to \mathcal{Y}$ which minimizes the probability that $y \neq H(x)$ on a newly observed example (x, y).

In the multi-label case, each instance $x \in \mathcal{X}$ may belong to multiple labels in $\mathcal{Y}$. Thus, a labeled example is a pair (x, Y) where $Y \subseteq \mathcal{Y}$ is the set of labels assigned to x. The single-label case is clearly a special case in which $|Y| = 1$ for all observations.

It is unclear in this setting precisely how to formalize the goal of a learning algorithm, and, in general, the "right" formalization may well depend on the problem at hand. One possibility is to seek a hypothesis which attempts to predict just one of the labels assigned to an example. In other words, the goal is to find $H : \mathcal{X} \to \mathcal{Y}$ which minimizes the probability that $H(x) \notin Y$ on a new observation (x, Y). We call this measure the *one-error* of hypothesis H since it measures the probability of not getting even one of the labels correct. We denote the one-error of a hypothesis h with respect to a distribution D over observations (x, Y) by one-err$_D(H)$. That is,

$$\text{one-err}_D(H) = \Pr_{(x,Y)\sim D}\left[H(x) \notin Y\right].$$

Note that, for single-label classification problems, the one-error is identical to ordinary error. In the following sections, we will introduce other loss measures that can be used in the multi-label setting, namely, Hamming loss and ranking loss. We also discuss modifications to AdaBoost appropriate to each case.

7. Using Hamming Loss for Multiclass Problems

Suppose now that the goal is to predict all and only all of the correct labels. In other words, the learning algorithm generates a hypothesis which predicts sets of labels, and the loss depends on how this predicted set differs from the one that was observed. Thus, $H : \mathcal{X} \to 2^{\mathcal{Y}}$ and, with respect to a distribution D, the loss is

$$\frac{1}{k}\,\mathrm{E}_{(x,Y)\sim D}\left[\,|h(x)\,\Delta\,Y|\,\right]$$

Given: $(x_1, Y_1), \ldots, (x_m, Y_m)$ where $x_i \in \mathcal{X}$, $Y_i \subseteq \mathcal{Y}$
Initialize $D_1(i, \ell) = 1/(mk)$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i, \ell) = \frac{D_t(i, \ell) \exp(-\alpha_t Y_i[\ell] h_t(x_i, \ell))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x, \ell) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x, \ell)\right).$$

Figure 2. AdaBoost.MH: A multiclass, multi-label version of AdaBoost based on Hamming loss.

where Δ denotes symmetric difference. (The leading $1/k$ is meant merely to ensure a value in $[0, 1]$.) We call this measure the *Hamming loss* of H, and we denote it by $\mathrm{hloss}_D(H)$.

To minimize Hamming loss, we can, in a natural way, decompose the problem into k orthogonal binary classification problems. That is, we can think of Y as specifying k binary labels (depending on whether a label y is or is not included in Y). Similarly, $h(x)$ can be viewed as k binary predictions. The Hamming loss then can be regarded as an average of the error rate of h on these k binary problems.

For $Y \subseteq \mathcal{Y}$, let us define $Y[\ell]$ for $\ell \in \mathcal{Y}$ to be

$$Y[\ell] = \begin{cases} +1 & \text{if } \ell \in Y \\ -1 & \text{if } \ell \notin Y. \end{cases}$$

To simplify notation, we also identify any function $H : \mathcal{X} \to 2^{\mathcal{Y}}$ with a corresponding two-argument function $H : \mathcal{X} \times \mathcal{Y} \to \{-1, +1\}$ defined by $H(x, \ell) = H(x)[\ell]$.

With the above reduction to binary classification in mind, it is rather straightforward to see how to use boosting to minimize Hamming loss. The main idea of the reduction is simply to replace each training example (x_i, Y_i) by k examples $((x_i, \ell), Y_i[\ell])$ for $\ell \in \mathcal{Y}$. The result is a boosting algorithm called AdaBoost.MH (shown in Figure 2) which maintains a distribution over examples i and labels ℓ. On round t, the weak learner accepts such a distribution D_t (as well as the training set), and generates a weak hypothesis $h_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. This reduction also leads to the choice of final hypothesis shown in the figure.

The reduction used to derive this algorithm combined with Theorem 1 immediately implies a bound on the Hamming loss of the final hypothesis:

THEOREM 3 *Assuming the notation of Figure 2, the following bound holds for the Hamming loss of H on the training data:*

$$\text{hloss}(H) \leq \prod_{t=1}^{T} Z_t.$$

We now can apply the ideas in the preceding sections to this binary classification problem. As before, our goal is to minimize

$$Z_t = \sum_{i,\ell} D_t(i,\ell) \exp(-\alpha_t \, Y_i[\ell] \, h_t(x_i,\ell)) \tag{12}$$

on each round. (Here, it is understood that the sum is over all examples indexed by i and all labels $\ell \in \mathcal{Y}$.)

As in Section 3.1, if we require that each h_t have range $\{-1,+1\}$ then we should choose

$$\alpha_t = \tfrac{1}{2} \ln \left(\frac{1+r_t}{1-r_t} \right) \tag{13}$$

where

$$r_t = \sum_{i,\ell} D_t(i,\ell) \, Y_i[\ell] \, h_t(x_i,\ell). \tag{14}$$

This gives

$$Z_t = \sqrt{1 - r_t^2}$$

and the goal of the weak learner becomes maximization of $|r_t|$.

Note that $(1-r_t)/2$ is equal to

$$\Pr_{(i,\ell)\sim D_t} \left[h_t(x_i,\ell) \neq Y_i[\ell] \right]$$

which can be thought of as a weighted Hamming loss with respect to D_t.

Example. As an example of how to maximize $|r_t|$, suppose our goal is to find an *oblivious* weak hypothesis h_t which ignores the instance x and predicts only on the basis of the label ℓ. Thus we can omit the x argument and write $h_t(x,\ell) = h_t(\ell)$. Let us also omit t subscripts. By symmetry, minimizing $-r$ is equivalent to maximizing r. So, we only need to find h which maximizes

$$\begin{aligned} r &= \sum_{i,\ell} D(i,\ell) \, Y_i[\ell] \, h(\ell) \\ &= \sum_{\ell} \left[h(\ell) \sum_i D(i,\ell) \, Y_i[\ell] \right]. \end{aligned}$$

Clearly, this is maximized by setting

$$h(\ell) = \text{sign} \left(\sum_i D(i,\ell) \, Y_i[\ell] \right).$$

■

7.1. *Domain-partitioning weak hypotheses*

We also can combine these ideas with those in Section 4.1 on domain-partitioning weak hypotheses. As in Section 4.1, suppose that h is associated with a partition $X_1, \ldots, X_N$ of the space $\mathcal{X}$. It is natural then to create partitions of the form $\mathcal{X} \times \mathcal{Y}$ consisting of all sets $X_j \times \{\ell\}$ for $j = 1, \ldots, N$ and $\ell \in \mathcal{Y}$. An appropriate hypothesis h can then be formed which predicts $h(x, \ell) = c_{j\ell}$ for $x \in X_j$. According to the results of Section 4.1, we should choose

$$c_{j\ell} = \frac{1}{2} \ln \left(\frac{W_+^{j\ell}}{W_-^{j\ell}} \right) \tag{15}$$

where $W_b^{j\ell} = \sum_i D(i, \ell) [\![x_i \in X_j \wedge Y_i[\ell] = b]\!]$. This gives

$$Z = 2 \sum_j \sum_\ell \sqrt{W_+^{j\ell} W_-^{j\ell}}. \tag{16}$$

7.2. *Relation to one-error and single-label classification*

We can use these algorithms even when the goal is to minimize one-error. The most natural way to do this is to set

$$H^1(x) = \arg\max_y \sum_t \alpha_t h_t(x, y), \tag{17}$$

i.e., to predict the label y most predicted by the weak hypotheses. The next simple theorem relates the one-error of H^1 and the Hamming loss of H.

THEOREM 4 *With respect to any distribution D over observations (x, Y) where $Y \neq \emptyset$,*

$$\text{one-err}_D(H^1) \leq k \, \text{hloss}_D(H).$$

Proof: Assume $Y \neq \emptyset$ and suppose $H^1(x) \notin Y$. We argue that this implies $H(x) \neq Y$. If the maximum in Eq. (17) is positive, then $H^1(x) \in H(x) - Y$. Otherwise, if the maximum is nonpositive, then $H(x) = \emptyset \neq Y$. In either case, $H(x) \neq Y$, i.e., $|H(x) \, \Delta \, Y| \geq 1$. Thus,

$$[\![H^1(x) \notin Y]\!] \leq |H(x) \, \Delta \, Y|$$

which, taking expectations, implies the theorem. ∎

In particular, this means that AdaBoost.MH can be applied to single-label multiclass classification problems. The resulting bound on the training error of the final hypothesis H^1 is at most

$$k \prod_t Z_t \tag{18}$$

where Z_t is as in Eq. (12). In fact, the results of Section 8 will imply a better bound of

$$\frac{k}{2} \prod_t Z_t. \tag{19}$$

Moreover, the leading constant $k/2$ can be improved somewhat by assuming without loss of generality that, prior to examining any of the data, a 0th weak hypothesis is chosen, namely $h_0 \equiv -1$. For this weak hypothesis, $r_0 = (k-2)/k$ and Z_0 is minimized by setting $\alpha_0 = \frac{1}{2}\ln(k-1)$ which gives $Z_0 = 2\sqrt{k-1}/k$. Plugging into the bound of Eq. (19), we therefore get an improved bound of

$$\frac{k}{2} \prod_{t=0}^{T} Z_t = \sqrt{k-1} \prod_{t=1}^{T} Z_t.$$

This hack is equivalent to modifying the algorithm of Figure 2 only in the manner in which D_1 is initialized. Specifically, D_1 should be chosen so that $D_1(i, y_i) = 1/(2m)$ (where y_i is the correct label for x_i) and $D_1(i, \ell) = 1/(2m(k-1))$ for $\ell \neq y_i$. Note that H^1 is unaffected.

8. Using Output Coding for Multiclass Problems

The method above maps a single-label problem into a multi-label problem in the simplest and most obvious way, namely, by mapping each single-label observation (x, y) to a multi-label observation $(x, \{y\})$. However, it may be more effective to use a more sophisticated mapping. In general, we can define a one-to-one mapping $\lambda : \mathcal{Y} \to 2^{\mathcal{Y}'}$ which we can use to map each observation (x, y) to $(x, \lambda(y))$. Note that λ maps to subsets of an unspecified label set $\mathcal{Y}'$ which need not be the same as $\mathcal{Y}$. Let $k' = |\mathcal{Y}'|$.

It is desirable to choose λ to be a function which maps different labels to sets which are far from one another, say, in terms of their symmetric difference. This is essentially the approach advocated by Dietterich and Bakiri (1995) in a somewhat different setting.They suggested using error correcting codes which are designed to have exactly this property. Alternatively, when k' is not too small, we can expect to get a similar effect by choosing λ entirely at random (so that, for $y \in \mathcal{Y}$ and $\ell \in \mathcal{Y}'$, we include or do not include ℓ in $\lambda(y)$ with equal probability). Once a function λ has been chosen we can apply AdaBoost.MH directly on the transformed training data $(x_i, \lambda(y_i))$.

How then do we classify a new instance x? The most direct use of Dietterich and Bakiri's approach is to evaluate H on x to obtain a set $H(x) \subseteq \mathcal{Y}'$. We then choose the label $y \in \mathcal{Y}$ for which the mapped output code $\lambda(y)$ has the shortest Hamming distance to $H(x)$. That is, we choose

$$\arg\min_{y \in \mathcal{Y}} |\lambda(y) \,\Delta\, H(x)|.$$

A weakness of this approach is that it ignores the confidence with which each label was included or not included in $H(x)$. An alternative approach is to predict that label y which, if it had been paired with x in the training set, would have caused (x, y) to be given the smallest weight under the final distribution. In other words, we suggest predicting the label

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$
a mapping $\lambda : \mathcal{Y} \to 2^{\mathcal{Y}'}$

- Run AdaBoost.MH on relabeled data: $(x_1, \lambda(y_1)), \ldots, (x_m, \lambda(y_m))$
- Get back final hypothesis H of form $H(x, y') = \mathrm{sign}(f(x, y'))$
  where $f(x, y') = \sum_t \alpha_t h_t(x, y')$
- Output modified final hypothesis:

$$\text{(Variant 1)} \quad H_1(x) = \arg\min_{y \in \mathcal{Y}} |\lambda(y) \,\Delta\, H(x)|$$

$$\text{(Variant 2)} \quad H_2(x) = \arg\min_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}'} \exp\left(-\lambda(y)[y']\, f(x, y')\right)$$

Figure 3. AdaBoost.MO: A multiclass version of AdaBoost based on output codes.

$$\arg\min_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}'} \exp(-\lambda(y)[y']\, f(x, y'))$$

where, as before, $f(x, y') = \sum_t \alpha_t h_t(x, y')$.

We call this version of boosting using output codes AdaBoost.MO. Pseudocode is given in Figure 3. The next theorem formalizes the intuitions above, giving a bound on training error in terms of the quality of the code as measured by the minimum distance between any pair of "code words."

THEOREM 5 *Assuming the notation of Figure 3 and Figure 2 (viewed as a subroutine), let*

$$\rho = \min_{\ell_1, \ell_2 \in \mathcal{Y} : \ell_1 \neq \ell_2} |\lambda(\ell_1) \,\Delta\, \lambda(\ell_2)|.$$

When run with this choice of λ, the training error of AdaBoost.MO is upper bounded by

$$\frac{2k'}{\rho} \prod_{t=1}^{T} Z_t$$

for Variant 1, and by

$$\frac{k'}{\rho} \prod_{t=1}^{T} Z_t$$

for Variant 2.

Proof: We start with Variant 1. Suppose the modified output hypothesis H_1 for Variant 1 makes a mistake on some example (x, y). This means that for some $\ell \neq y$,

$$|H(x) \,\Delta\, \lambda(\ell)| \leq |H(x) \,\Delta\, \lambda(y)|$$

which implies that

$$\begin{aligned}
2|H(x)\,\Delta\,\lambda(y)| &\geq |H(x)\,\Delta\,\lambda(y)| + |H(x)\,\Delta\,\lambda(\ell)| \\
&\geq |(H(x)\Delta\lambda(y))\,\Delta\,(H(x)\Delta\lambda(\ell))| \\
&= |\lambda(y)\,\Delta\,\lambda(\ell)| \\
&\geq \rho
\end{aligned}$$

where the second inequality uses the fact that $|A\,\Delta\,B| \leq |A| + |B|$ for any sets A and B. Thus, in case of an error, $|H(x)\,\Delta\,\lambda(y)| \geq \rho/2$. On the other hand, the Hamming error of AdaBoost.MH on the training set is, by definition,

$$\frac{1}{mk'}\sum_{i=1}^{m}|H(x_i)\,\Delta\,\lambda(y_i)|$$

which is at most $\prod_t Z_t$ by Theorem 3. Thus, if M is the number of training mistakes, then

$$M\frac{\rho}{2} \leq \sum_{i=1}^{m}|H(x_i)\,\Delta\,\lambda(y_i)| \leq mk'\prod_t Z_t$$

which implies the stated bound.

For Variant 2, suppose that H_2 makes an error on some example (x, y). Then for some $\ell \neq y$

$$\sum_{y'\in\mathcal{Y}'}\exp\left(-\lambda(\ell)[y']\,f(x,y')\right) \leq \sum_{y'\in\mathcal{Y}'}\exp\left(-\lambda(y)[y']\,f(x,y')\right). \tag{20}$$

Fixing x, y and ℓ, let us define $w(y') = \exp\left(-\lambda(y)[y']\,f(x,y')\right)$. Note that

$$\exp\left(-\lambda(\ell)[y']\,f(x,y')\right) = \begin{cases} w(y') & \text{if } \lambda(y)[y'] = \lambda(\ell)[y'] \\ 1/w(y') & \text{otherwise.}\end{cases}$$

Thus, Eq. (20) implies that

$$\sum_{y'\in S} w(y') \geq \sum_{y'\in S} 1/w(y')$$

where $S = \lambda(y)\Delta\lambda(\ell)$. This implies that

$$\sum_{y'\in\mathcal{Y}'} w(y') \geq \sum_{y'\in S} w(y') \geq \tfrac{1}{2}\sum_{y'\in S}(w(y') + 1/w(y')) \geq |S| \geq \rho.$$

The third inequality uses the fact that $x + 1/x \geq 2$ for all $x > 0$. Thus, we have shown that if a mistake occurs on (x, y) then

$$\sum_{y'\in\mathcal{Y}'}\exp\left(-\lambda(y)[y']\,f(x,y')\right) \geq \rho.$$

If M is the number of training errors under Variant 2, this means that

$$\rho M \leq \sum_{i=1}^{m} \sum_{y' \in \mathcal{Y}'} \exp\left(-\lambda(y_i)[y']\, f(x_i, y')\right) = mk' \prod_t Z_t$$

where the equality uses the main argument of the proof of Theorem 1 combined with the reduction to binary classification described just prior to Theorem 3. This immediately implies the stated bound. ■

If the code λ is chosen at random (uniformly among all possible codes), then, for large k', we expect ρ to approach $(1/2 - o(1))k'$. In this case, the leading coefficients in the bounds of Theorem 5 approach 4 for Variant 1 and 2 for Variant 2, independent of the number of classes k in the original label set $\mathcal{Y}$.

We can use Theorem 5 to improve the bound in Eq. (18) for AdaBoost.MH to that in Eq. (19). We apply Theorem 5 to the code defined by $\lambda(y) = \{y\}$ for all $y \in \mathcal{Y}$. Clearly, $\rho = 2$ in this case. Moreover, we claim that H^1 as defined in Eq. (17) produces identical predictions to those generated by Variant 2 in AdaBoost.MO since

$$\sum_{y' \in \mathcal{Y}} \exp\left(-\lambda(y)[y']\, f(x, y')\right) = e^{-f(x,y)} - e^{f(x,y)} + \sum_{y' \in \mathcal{Y}} e^{f(x,y')}. \tag{21}$$

Clearly, the minimum of Eq. (21) over y is attained when $f(x, y)$ is maximized. Applying Theorem 5 now gives the bound in Eq. (19).

9. Using Ranking Loss for Multiclass Problems

In Section 7, we looked at the problem of finding a hypothesis that exactly identifies the labels associated with an instance. In this section, we consider a different variation of this problem in which the goal is to find a hypothesis which *ranks* the labels with the hope that the correct labels will receive the highest ranks. The approach described here is closely related to one used by Freund et al. (1998) for using boosting for more general ranking problems.

To be formal, we now seek a hypothesis of the form $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ with the interpretation that, for a given instance x, the labels in $\mathcal{Y}$ should be ordered according to $f(x, \cdot)$. That is, a label ℓ_1 is considered to be ranked higher than ℓ_2 if $f(x, \ell_1) > f(x, \ell_2)$. With respect to an observation (x, Y), we only care about the relative ordering of the *crucial pairs* ℓ_0, ℓ_1 for which $\ell_0 \notin Y$ and $\ell_1 \in Y$. We say that f *misorders* a crucial pair ℓ_0, ℓ_1 if $f(x, \ell_1) \leq f(x, \ell_0)$ so that f fails to rank ℓ_1 above ℓ_0. Our goal is to find a function f with a small number of misorderings so that the labels in Y are ranked above the labels not in Y.

Our goal then is to minimize the expected fraction of crucial pairs which are misordered. This quantity is called the *ranking loss*, and, with respect to a distribution D over observations, it is defined to be

$$\mathrm{E}_{(x,Y)\sim D}\left[\frac{|\{(\ell_0, \ell_1) \in (\mathcal{Y} - Y) \times Y : f(x, \ell_1) \leq f(x, \ell_0)\}|}{|Y|\,|\mathcal{Y} - Y|}\right].$$

We denote this measure $\mathrm{rloss}_D(f)$. Note that we assume that Y is never empty nor equal to all of $\mathcal{Y}$ for any observation since there is no ranking problem to be solved in this case.

Given: $(x_1, Y_1), \ldots, (x_m, Y_m)$ where $x_i \in \mathcal{X}$, $Y_i \subseteq \mathcal{Y}$

Initialize $D_1(i, \ell_0, \ell_1) = \begin{cases} 1/(m \cdot |Y_i| \cdot |\mathcal{Y} - Y_i|) & \text{if } \ell_0 \notin Y_i \text{ and } \ell_1 \in Y_i \\ 0 & \text{else.} \end{cases}$

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i, \ell_0, \ell_1) = \frac{D_t(i, \ell_0, \ell_1) \exp\left(\frac{1}{2}\alpha_t(h_t(x_i, \ell_0) - h_t(x_i, \ell_1))\right)}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$f(x, \ell) = \sum_{t=1}^{T} \alpha_t h_t(x, \ell).$$

Figure 4. AdaBoost.MR: A multiclass, multi-label version of AdaBoost based on ranking loss.

A version of AdaBoost for ranking loss called AdaBoost.MR is shown in Figure 4. We now maintain a distribution D_t over $\{1, \ldots, m\} \times Y \times Y$. This distribution is zero, however, except on the relevant triples (i, ℓ_0, ℓ_1) for which ℓ_0, ℓ_1 is a crucial pair relative to (x_i, Y_i).

Weak hypotheses have the form $h_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. We think of these as providing a ranking of labels as described above. The update rule is a bit new. Let ℓ_0, ℓ_1 be a crucial pair relative to (x_i, Y_i) (recall that D_t is zero in all other cases). Assuming momentarily that $\alpha_t > 0$, this rule decreases the weight $D_t(i, \ell_0, \ell_1)$ if h_t gives a correct ranking ($h_t(x_i, \ell_1) > h_t(x_i, \ell_0)$), and increases this weight otherwise.

We can prove a theorem analogous to Theorem 1 for ranking loss:

THEOREM 6 *Assuming the notation of Figure 4, the following bound holds for the ranking loss of f on the training data:*

$$\text{rloss}(f) \leq \prod_{t=1}^{T} Z_t.$$

Proof: The proof is very similar to that of Theorem 1.

Unraveling the update rule, we have that

$$D_{T+1}(i, \ell_0, \ell_1) \;=\; \frac{D_1(i, \ell_0, \ell_1) \exp\left(\frac{1}{2}(f(x_i, \ell_0) - f(x_i, \ell_1))\right)}{\prod_t Z_t}.$$

The ranking loss on the training set is

$$\begin{aligned}
&\sum_{i,\ell_0,\ell_1} D_1(i,\ell_0,\ell_1)[\![f(x_i,\ell_0) \geq f(x_i,\ell_1)]\!] \\
&\quad \leq \sum_{i,\ell_0,\ell_1} D_1(i,\ell_0,\ell_1) \exp\left(\tfrac{1}{2}(f(x_i,\ell_0) - f(x_i,\ell_1))\right) \\
&\quad = \sum_{i,\ell_0,\ell_1} D_{T+1}(i,\ell_0,\ell_1) \prod_t Z_t = \prod_t Z_t.
\end{aligned}$$

(Here, each of the sums is over all example indices i and all pairs of labels in $\mathcal{Y} \times \mathcal{Y}$.) This completes the theorem. ■

So, as before, our goal on each round is to try to minimize

$$Z = \sum_{i,\ell_0,\ell_1} D(i,\ell_0,\ell_1) \exp\left(\tfrac{1}{2}\alpha(h(x_i,\ell_0) - h(x_i,\ell_1))\right)$$

where, as usual, we omit t subscripts. We can apply all of the methods described in previous sections. Starting with the exact methods for finding α, suppose we are given a hypothesis h. Then we can make the appropriate modifications to the method of Section 3.2 to find α numerically.

Alternatively, in the special case that h has range $\{-1,+1\}$, we have that

$$\tfrac{1}{2}\left(h(x_i,\ell_0) - h(x_i,\ell_1)\right) \in \{-1, 0, +1\} \ .$$

Therefore, we can use the method of Section 3.3 to choose α exactly:

$$\alpha = \tfrac{1}{2} \ln\left(\frac{W_+}{W_-}\right) \tag{22}$$

where

$$W_b = \sum_{i,\ell_0,\ell_1} D(i,\ell_0,\ell_1)[\![h(x_i,\ell_0) - h(x_i,\ell_1) = 2b]\!]. \tag{23}$$

As before,

$$Z = W_0 + 2\sqrt{W_- W_+} \tag{24}$$

in this case.

How can we find a weak hypothesis to minimize this expression? A simplest first case is to try to find the best oblivious weak hypothesis. An interesting open problem then is, given a distribution D, to find an oblivious hypothesis $h : \mathcal{Y} \to \{-1,+1\}$ which minimizes Z when defined as in Eqs. (23) and (24). We suspect that this problem may be NP-complete when the size of $\mathcal{Y}$ is not fixed.

We also do not know how to analytically find the best oblivious hypothesis when we do not restrict the range of h, although numerical methods may be reasonable. Note that finding the best oblivious hypothesis is the simplest case of the natural extension of the technique of Section 4.1 to ranking loss. Folding $\alpha/2$ into h as in Section 4, the problem is to find $h : \mathcal{Y} \to \mathbb{R}$ to minimize

$$Z = \sum_{\ell_0,\ell_1} \left[\left(\sum_i D(i,\ell_0,\ell_1) \right) \exp(h(\ell_0) - h(\ell_1)) \right] .$$

This can be rewritten as

$$Z = \sum_{\ell_0,\ell_1} \left[w(\ell_0,\ell_1) \exp(h(\ell_0) - h(\ell_1)) \right] \tag{25}$$

where $w(\ell_0,\ell_1) = \sum_i D(i,\ell_0,\ell_1)$. In Appendix A we show that expressions of the form given by Eq. (25) are convex, and we discuss how to minimize such expressions. (To see that the expression in Eq. (25) has the general form of Eq. (A.1), identify the $w(\ell_0,\ell_1)$'s with the w_i's in Eq. (A.1), and the $h(\ell)$'s with the a_j's.)

Since exact analytic solutions seem hard to come by for ranking loss, we next consider approximations such as those in Section 3.1. Assuming weak hypotheses h with range in $[-1,+1]$, we can use the same approximation of Eq. (4) which yields

$$Z \le \left(\frac{1-r}{2} \right) e^{\alpha} + \left(\frac{1+r}{2} \right) e^{-\alpha} \tag{26}$$

where

$$r = \tfrac{1}{2} \sum_{i,\ell_0,\ell_1} D(i,\ell_0,\ell_1)(h(x_i,\ell_1) - h(x_i,\ell_0)). \tag{27}$$

As before, the right hand side of Eq. (26) is minimized when

$$\alpha = \tfrac{1}{2} \ln \left(\frac{1+r}{1-r} \right) \tag{28}$$

which gives

$$Z \le \sqrt{1-r^2}.$$

Thus, a reasonable and more tractable goal for the weak learner is to try to maximize $|r|$.

Example. To find the oblivious weak hypothesis $h : \mathcal{Y} \to \{-1,+1\}$ which maximizes r, note that by rearranging sums,

$$r = \sum_{\ell} h(\ell)\pi(\ell)$$

where

$$\pi(\ell) = \tfrac{1}{2} \sum_{i,\ell'} \left(D(i,\ell',\ell) - D(i,\ell,\ell') \right) .$$

Clearly, r is maximized if we set $h(\ell) = \mathrm{sign}(\pi(\ell))$. ■

Note that, although we use this approximation to find the weak hypothesis, once the weak hypothesis has been computed by the weak learner, we can use other methods to choose α such as those outlined above.

Given: $(x_1, Y_1), \ldots, (x_m, Y_m)$ where $x_i \in \mathcal{X}$, $Y_i \subseteq \mathcal{Y}$
Initialize $v_1(i, \ell) = (m \cdot |Y_i| \cdot |\mathcal{Y} - Y_i|)^{-1/2}$
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t (as defined by Eq. (29))
- Get weak hypothesis $h_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$v_{t+1}(i, \ell) = \frac{v_t(i, \ell) \exp\left(-\frac{1}{2}\alpha_t Y_i[\ell] h_t(x_i, \ell)\right)}{\sqrt{Z_t}}$$

where

$$Z_t = \sum_i \left[\left(\sum_{\ell \notin Y_i} v_t(i, \ell) \exp\left(\tfrac{1}{2}\alpha_t h_t(x_i, \ell)\right) \right) \left(\sum_{\ell \in Y_i} v_t(i, \ell) \exp\left(-\tfrac{1}{2}\alpha_t h_t(x_i, \ell)\right) \right) \right]$$

Output the final hypothesis:

$$f(x, \ell) = \sum_{t=1}^{T} \alpha_t h_t(x, \ell).$$

Figure 5. A more efficient version of AdaBoost.MR (Figure 4).

9.1. A more efficient implementation

The method described above may be time and space inefficient when there are many labels. In particular, we naively need to maintain $|Y_i| \cdot |\mathcal{Y} - Y_i|$ weights for each training example (x_i, Y_i), and each weight must be updated on each round. Thus, the space complexity and time-per-round complexity can be as bad as $\theta(mk^2)$.

In fact, the same algorithm can be implemented using only $O(mk)$ space and time per round. By the nature of the updates, we will show that we only need to maintain weights v_t over $\{1, \ldots, m\} \times \mathcal{Y}$. We will maintain the condition that if ℓ_0, ℓ_1 is a crucial pair relative to (x_i, Y_i), then

$$D_t(i, \ell_0, \ell_1) = v_t(i, \ell_0) \cdot v_t(i, \ell_1) \tag{29}$$

at all times. (Recall that D_t is zero for all other triples (i, ℓ_0, ℓ_1).)

The pseudocode for this implementation is shown in Figure 5. Eq. (29) can be proved by induction. It clearly holds initially. Using our inductive hypothesis, it is straightforward to expand the computation of Z_t in Figure 5 to see that it is equivalent to the computation of Z_t in Figure 4. To show that Eq. (29) holds on round $t + 1$, we have, for crucial pair ℓ_0, ℓ_1:

$$D_{t+1}(i, \ell_0, \ell_1) \;=\; \frac{D_t(i, \ell_0, \ell_1) \exp\left(\frac{1}{2}\alpha_t (h_t(x_i, \ell_0) - h_t(x_i, \ell_1))\right)}{Z_t}$$

$$\begin{aligned} &= \frac{v_t(i,\ell_0)\exp\left(\frac{1}{2}\alpha_t h_t(x_i,\ell_0)\right)}{\sqrt{Z_t}} \cdot \frac{v_t(i,\ell_1)\exp\left(-\frac{1}{2}\alpha_t h_t(x_i,\ell_1)\right)}{\sqrt{Z_t}} \\ &= v_{t+1}(i,\ell_0) \cdot v_{t+1}(i,\ell_1)\ . \end{aligned}$$

Finally, note that all space requirements and all per-round computations are $O(mk)$, with the possible exception of the call to the weak learner. However, if we want the weak learner to maximize $|r|$ as in Eq. (27), then we also only need to pass mk weights to the weak learner, all of which can be computed in $O(mk)$ time. Omitting t subscripts, we can rewrite r as

$$\begin{aligned} r &= \frac{1}{2} \sum_{i,\ell_0,\ell_1} D(i,\ell_0,\ell_1)(h(x_i,\ell_1) - h(x_i,\ell_0)) \\ &= \frac{1}{2} \sum_i \sum_{\ell_0 \notin Y_i, \ell_1 \in Y_i} v(i,\ell_0) v(i,\ell_1)\left(h(x_i,\ell_1) Y_i[\ell_1] + h(x_i,\ell_0) Y_i[\ell_0]\right) \\ &= \frac{1}{2} \sum_i \left[\sum_{\ell_0 \notin Y_i} \left(v(i,\ell_0) \sum_{\ell_1 \in Y_i} v(i,\ell_1) \right) Y_i[\ell_0]\, h(x_i,\ell_0)\ + \right. \\ &\qquad \left. \sum_{\ell_1 \in Y_i} \left(v(i,\ell_1) \sum_{\ell_0 \notin Y_i} v(i,\ell_0) \right) Y_i[\ell_1]\, h(x_i,\ell_1) \right] \\ &= \sum_{i,\ell} d(i,\ell)\, Y_i[\ell]\, h(x_i,\ell) \end{aligned} \tag{30}$$

where

$$d(i,\ell) = \frac{1}{2} v(i,\ell) \sum_{\ell' : Y_i[\ell'] \neq Y_i[\ell]} v(i,\ell')\ .$$

All of the weights $d(i,\ell)$ can be computed in $O(mk)$ time by first computing the sums which appear in this equation for the two possible cases that $Y_i[\ell]$ is -1 or $+1$. Thus, we only need to pass $O(mk)$ weights to the weak learner in this case rather than the full distribution D_t of size $O(mk^2)$. Moreover, note that Eq. (30) has exactly the same form as Eq. (14) which means that, in this setting, the same weak learner can be used for either Hamming loss or ranking loss.

9.2. *Relation to one-error*

As in Section 7.2, we can use the ranking loss method for minimizing one-error, and therefore also for single-label problems. Indeed, Freund and Schapire's (1997) "pseudoloss"-based algorithm AdaBoost.M2 is a special case of the use of ranking loss in which all data are single-labeled, the weak learner attempts to maximize $|r_t|$ as in Eq. (27), and α_t is set as in Eq. (28).

As before, the natural prediction rule is

$$H^1(x) = \arg\max_y \sum_t f(x,y),$$

in other words, to choose the highest ranked label for instance x. We can show:

THEOREM 7 *With respect to any distribution D over observations (x, Y) where Y is neither empty nor equal to $\mathcal{Y}$,*

$$\text{one-err}_D(H^1) \leq (k-1)\,\text{rloss}_D(f).$$

Proof: Suppose $H^1(x) \notin Y$. Then, with respect to f and observation (x, Y), misorderings occur for all pairs $\ell_1 \in Y$ and $\ell_0 = H^1(x)$. Thus,

$$\frac{|\{(\ell_0, \ell_1) \in (\mathcal{Y} - Y) \times Y : f(x, \ell_1) \leq f(x, \ell_0)\}|}{|Y| \cdot |\mathcal{Y} - Y|} \;\geq\; \frac{1}{|\mathcal{Y} - Y|} \;\geq\; \frac{1}{k-1}\,.$$

Taking expectations gives

$$\frac{1}{k-1}\mathrm{E}_{(x,Y)\sim D}\left[\; [\![H^1(x) \notin Y]\!] \;\right] \leq \text{rloss}_D(f)$$

which proves the theorem. ∎

10. Experiments

In this section, we describe a few experiments that we ran on some of the boosting algorithms described in this paper. The first set of experiments compares the algorithms on a set of learning benchmark problems from the UCI repository. The second experiment does a comparison on a large text categorization task. More details of our text-categorization experiments appear in a companion paper (Schapire & Singer, to appear).

For multiclass problems, we compared three of the boosting algorithms:

Discrete AdaBoost.MH: In this version of AdaBoost.MH, we require that weak hypotheses have range $\{-1, +1\}$. As described in Section 7, we set α_t as in Eq. (13). The goal of the weak learner in this case is to maximize $|r_t|$ as defined in Eq. (14).

Real AdaBoost.MH: In this version of AdaBoost.MH, we do not restrict the range of the weak hypotheses. Since all our experiments involve domain-partitioning weak hypotheses, we can set the confidence-ratings as in Section 7.1 (thereby eliminating the need to choose α_t). The goal of the weak learner in this case is to minimize Z_t as defined in Eq. (16). We also smoothed the predictions as in Sec. 4.2 using $\varepsilon = 1/(2mk)$.

Discrete AdaBoost.MR: In this version of AdaBoost.MR, we require that weak hypotheses have range $\{-1, +1\}$. We use the approximation of Z_t given in Eq. (26) and therefore set α_t as in Eq. (28) with a corresponding goal for the weak learner of maximizing $|r_t|$ as defined in Eq. (27). Note that, in the single-label case, this algorithm is identical to Freund and Schapire's (1997) AdaBoost.M2 algorithm.

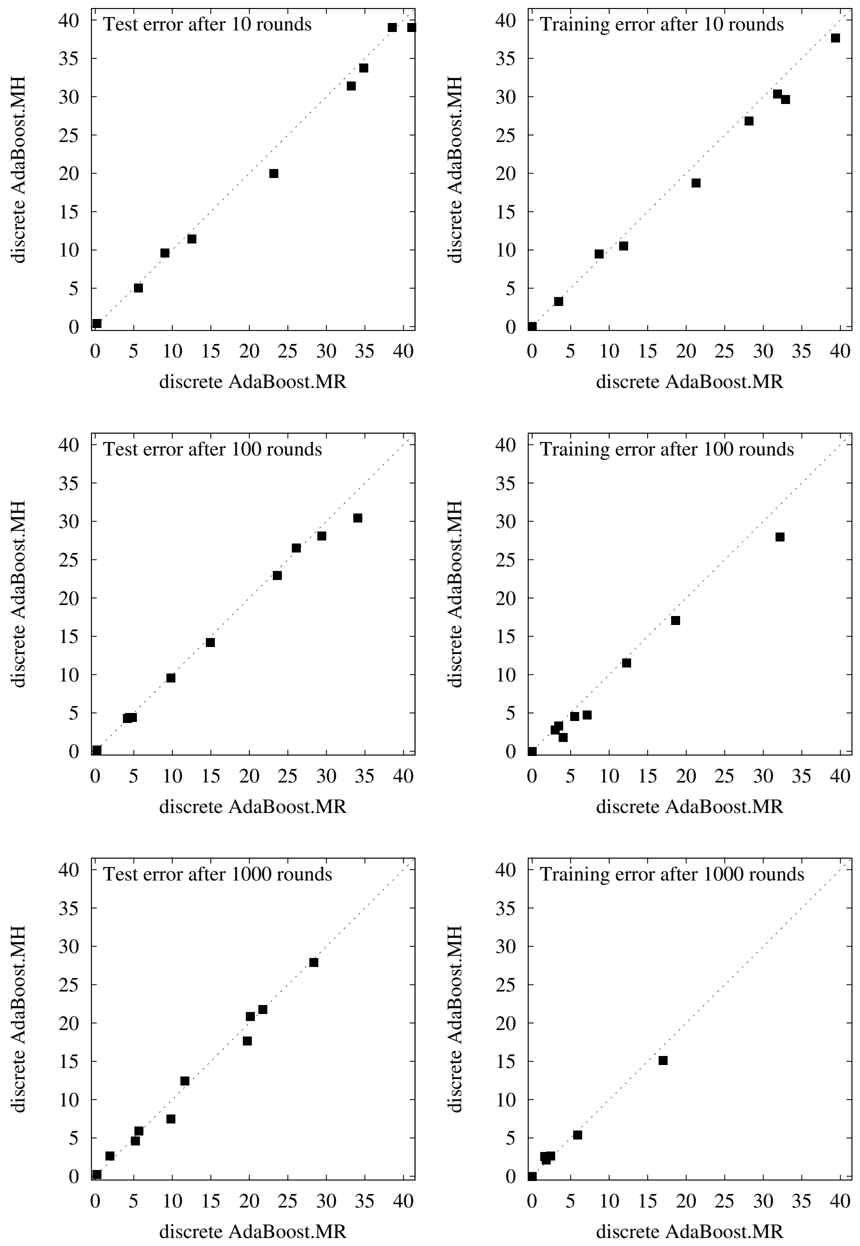

Figure 6. Comparison of discrete AdaBoost.MH and discrete AdaBoost.MR on 11 multiclass benchmark problems from the UCI repository. Each point in each scatterplot shows the error rate of the two competing algorithms on a single benchmark. Top and bottom rows give training and test errors, respectively, for 10, 100 and 1000 rounds of boosting. (However, on one benchmark dataset, the error rates fell outside the given range when only 10 rounds of boosting were used.)

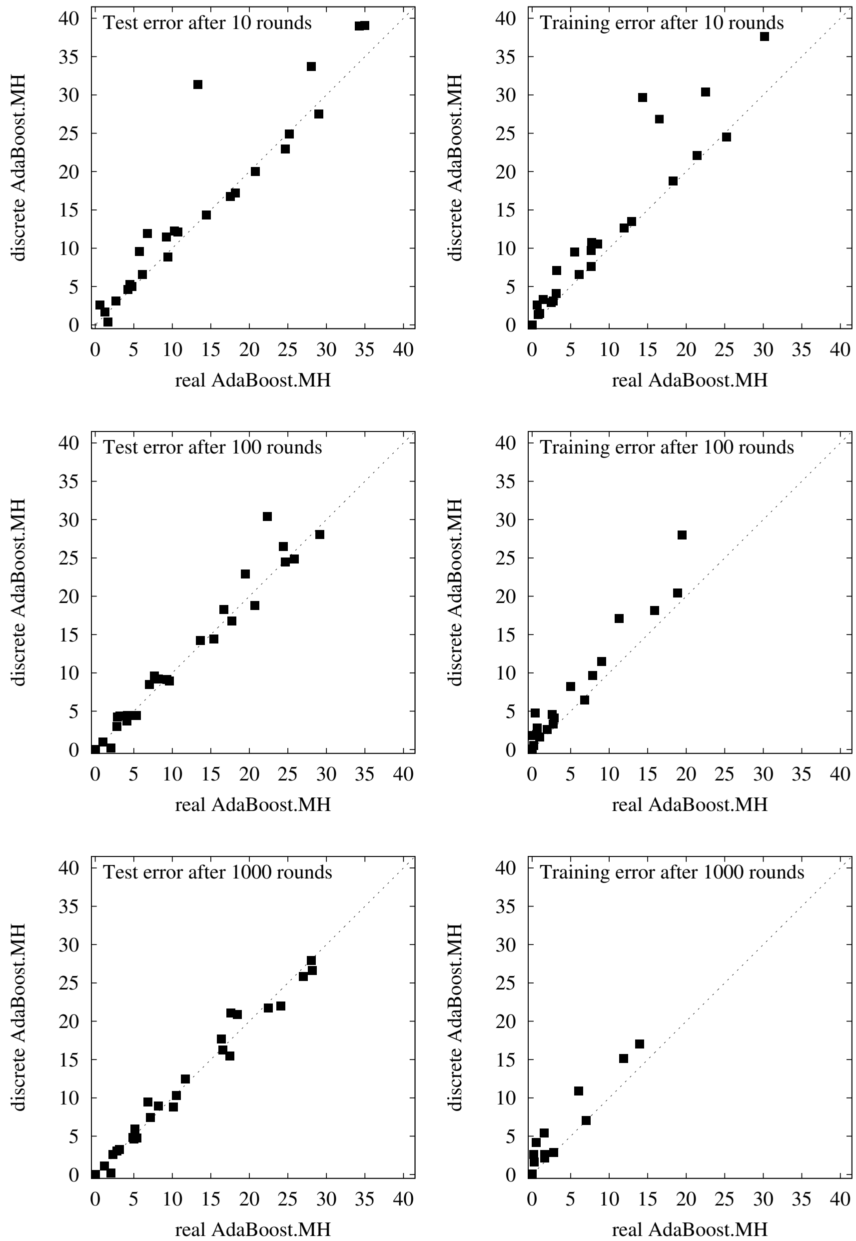

Figure 7. Comparison of discrete and real AdaBoost.MH on 26 binary and multiclass benchmark problems from the UCI repository. (See caption for Figure 6.)

We used these algorithms for two-class and multiclass problems alike. Note, however, that discrete AdaBoost.MR and discrete AdaBoost.MH are equivalent algorithms for two-class problems.

We compared the three algorithms on a collection of benchmark problems available from the repository at University of California at Irvine (Merz & Murphy, 1998). We used the same experimental set-up as Freund and Schapire (1996). Namely, if a test set was already provided, experiments were run 20 times and the results averaged (since some of the learning algorithms may be 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. We tested on the same set of benchmarks, except that we dropped the "vowel" dataset. Each version of AdaBoost was run for 1000 rounds.

We used the simplest of the weak learners tested by Freund and Schapire (1996). This weak learner finds a weak hypothesis which 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. Such a hypothesis can be viewed as a one-level decision tree (sometimes called a "decision stump"). The best hypothesis of this form which optimizes the appropriate learning criterion (as listed above) can always be found by a direct and efficient search using the methods described in this paper.

Figure 6 compares the relative performance of Freund and Schapire's AdaBoost.M2 algorithm (here called "discrete AdaBoost.MR") to the new algorithm, discrete AdaBoost.MH. Each point in each scatterplot gives the (averaged) error rates of the two methods for a single benchmark problem; that is, the x-coordinate of a point gives the error rate for discrete AdaBoost.MR, and the y-coordinate gives the error rate for discrete AdaBoost.MH. (Since the two methods are equivalent for two-class problems, we only give results for the multiclass benchmarks.) We have provided scatterplots for 10, 100 and 1000 rounds of boosting, and for test and train error rates. It seems rather clear from these figures that the two methods are generally quite evenly matched with a possible slight advantage to AdaBoost.MH. Thus, for these problems, the Hamming loss methodology gives comparable results to Freund and Schapire's method, but has the advantage of being conceptually simpler.

Next, we assess the value of using weak hypotheses which give confidence-rated predictions. Figure 7 shows similar scatterplots comparing real AdaBoost.MH and discrete AdaBoost.MH. These scatterplots show that the real version (with confidences) is overall more effective at driving down the training error, and also has an advantage on the test error rate, especially for a relatively small number of rounds. By 1000 rounds, however, these differences largely disappear.

In Figures 8 and 9, we give more details on the behavior of the different versions of AdaBoost. In Figure 8, we compare discrete and real AdaBoost.MH on 16 different problems from the UCI repository. For each problem we plot for each method its training and test error as a function of the number of rounds of boosting. Similarly, in Figure 8 we compare discrete AdaBoost.MR, discrete AdaBoost.MH, and real AdaBoost.MH on multiclass problems.

After examining the behavior of the various error curves, the potential for improvement of AdaBoost with real-valued predictions seems to be greatest on larger problems. The most noticeable case is the "letter-recognition" task, the largest UCI problem in our suite.

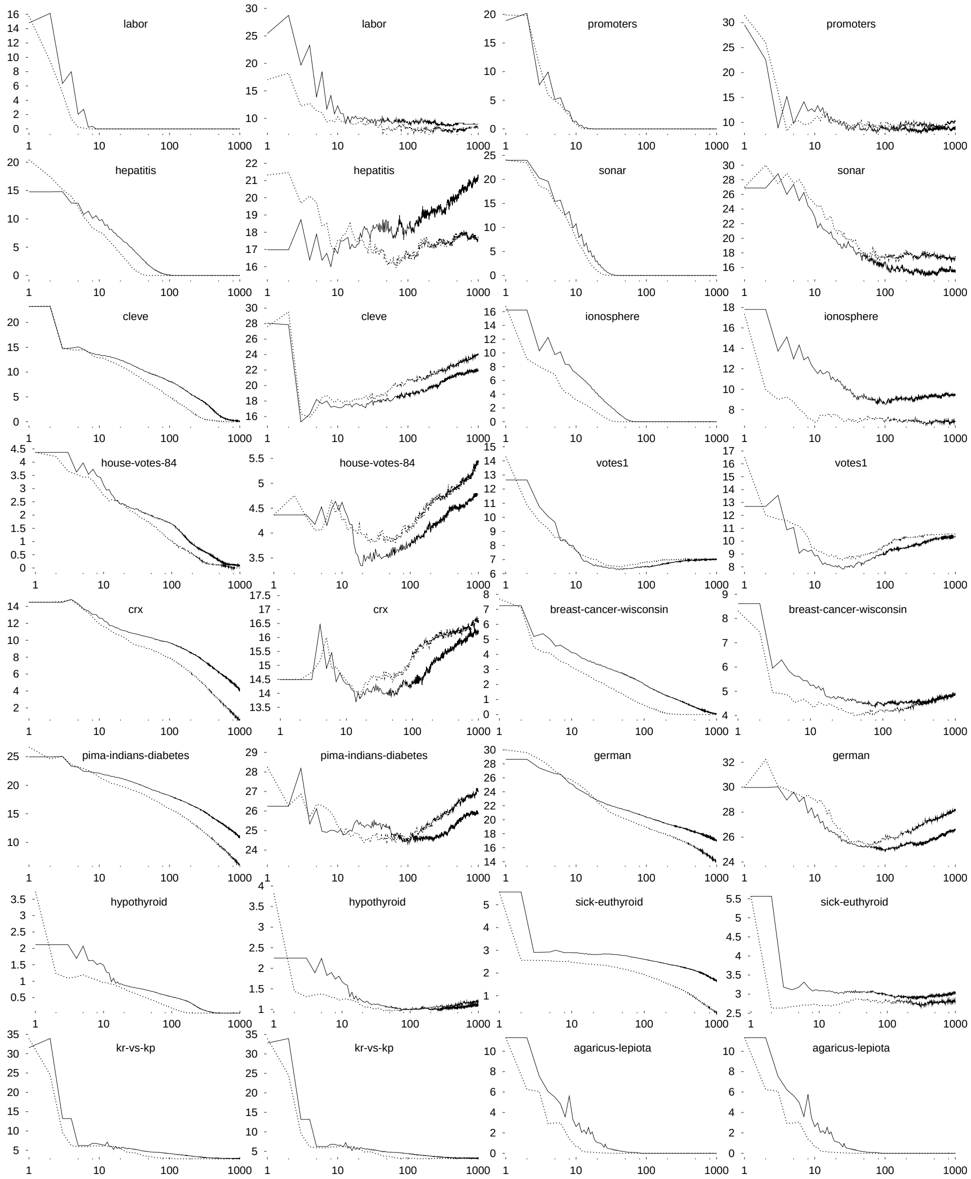

Figure 8. Comparison of discrete and real AdaBoost.MH on 16 binary problems from UCI repository. For each problem we show the training (left) and test (right) errors as a function of number of rounds of boosting.

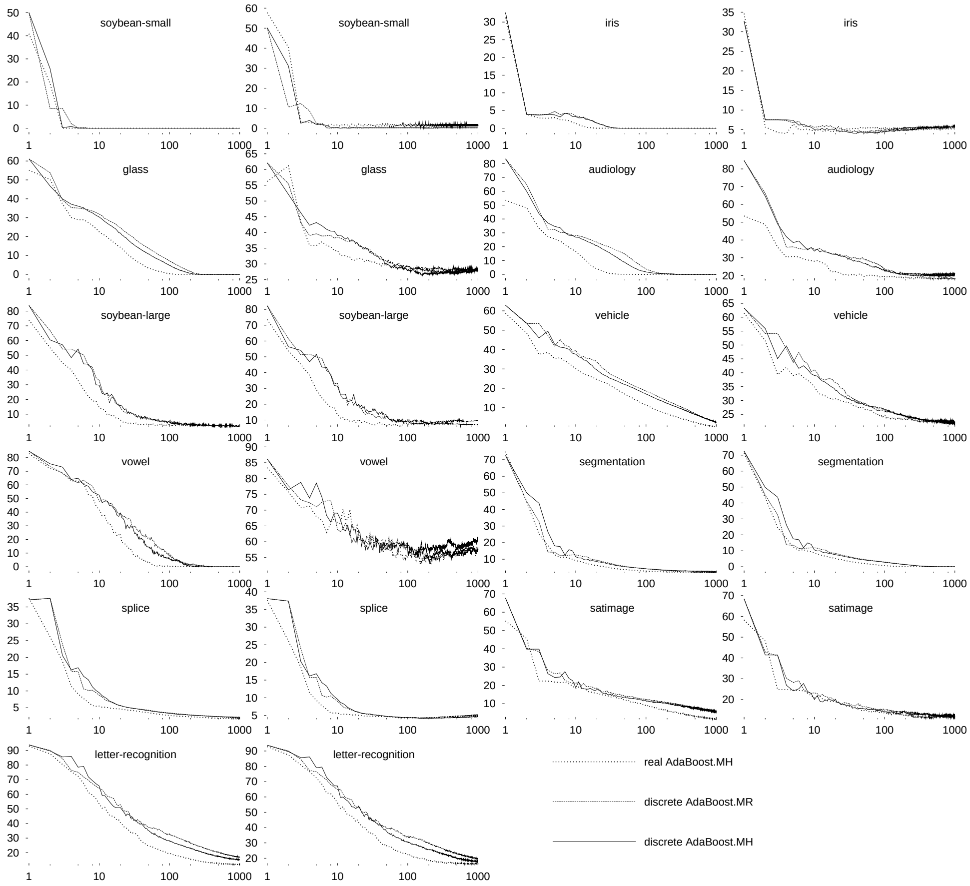

Figure 9. Comparison of discrete AdaBoost.MR, discrete AdaBoost.MH, and real AdaBoost.MH on 11 multiclass problems from UCI repository. For each problem we show the training (left) and test (right) errors as a function of number of rounds of boosting.

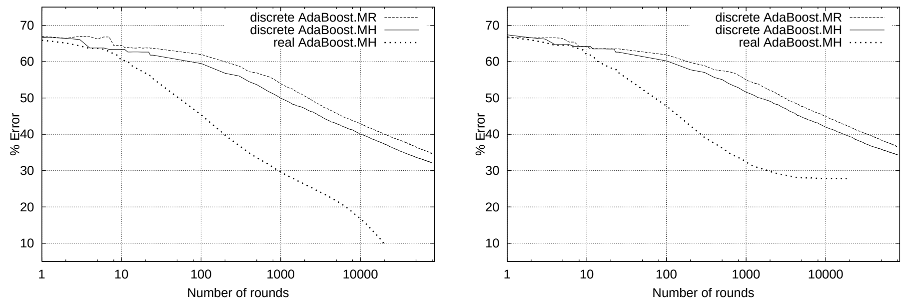

Figure 10. Comparison of the training (left) and test (right) error using three boosting methods on a six-class text classification problem from the TREC-AP collection.

This is a 26-class problem with $16,000$ training examples and $4,000$ test examples. For this problem, the training error after 100 rounds is 32.2% for discrete AdaBoost.MR, 28.0% for discrete AdaBoost.MH, and 19.5% for real AdaBoost.MH. The test error rates after 100 rounds are 34.1%, 30.4% and 22.3%, respectively. By 1,000 rounds, this gap in test error has narrowed somewhat to 19.7%, 17.6% and 16.4%.

Finally, we give results for a large text-categorization problem. More details of our text-categorization experiments are described in a companion paper (Schapire & Singer, to appear). In this problem, there are six classes: DOMESTIC, ENTERTAINMENT, FINANCIAL, INTERNATIONAL, POLITICAL, WASHINGTON. The goal is to assign a document to one, and only one, of the above classes. We use the same weak learner as above, appropriately modified for text; specifically, the weak hypotheses make their predictions based on tests that check for the presence or absence of a phrase in a document. There are 142,727 training documents and 66,973 test documents.

In Figure 10, we compare the performance of discrete AdaBoost.MR, discrete AdaBoost.MH and real AdaBoost.MH. The figure shows the training and test error as a function of number of rounds. The x-axis shows the number of rounds (using a logarithmic scale), and the y-axis the training and test error. Real AdaBoost.MH dramatically outperforms the other two methods, a behavior that seems to be typical on large text-categorization tasks. For example, to reach a test error of 40%, discrete AdaBoost.MH takes 16,938 rounds, and discrete AdaBoost.MR takes 33,347 rounds. In comparison, real AdaBoost.MH takes only 268 rounds, more than a sixty-fold speed-up over the best of the other two methods!

As happened in this example, discrete AdaBoost.MH seems to consistently outperform discrete AdaBoost.MR on similar problems. However, this might be partially due to the inferior choice of α_t using the approximation leading to Eq. (28) rather than the exact method which gives the choice of α_t in Eq. (22).

11. Concluding Remarks

In this paper, we have described several improvements to Freund and Schapire's AdaBoost algorithm. In the new framework, weak hypotheses may assign confidences to each of their predictions. We described several generalizations for multiclass problems. The experimental results with the improved boosting algorithms show that dramatic improvements in training error are possible when a fairly large amount of data is available. However, on small and noisy datasets, the rapid decrease of training error is often accompanied with overfitting which sometimes results in rather poor generalization error. A very important research goal is thus to control, either directly or indirectly, the complexity of the strong hypotheses constructed by boosting.

Several applications can make use of the improved boosting algorithms. We have implemented a system called BoosTexter for multiclass multi-label text and speech categorization and performed an extensive set of experiments with this system (Schapire & Singer, to appear). We have also used the new boosting framework for devising efficient ranking algorithms (Freund et al., 1998).

There are other domains that may make use of the new framework for boosting. For instance, it might be possible to train non-linear classifiers, such as neural networks using Z as the objective function. We have also mentioned several open problems such as finding an oblivious hypothesis into $\{-1, +1\}$ which minimizes Z in AdaBoost.MR.

Finally, there seem to be interesting connections between boosting and other models and their learning algorithms such as generalized additive models (Friedman et al., 1998) and maximum entropy methods (Csiszár & Tusnády, 1984) which form a new and exciting research arena.

Acknowledgments

We would like to thank Yoav Freund and Raj Iyer for many helpful discussions. Thanks also to Peter Bartlett for showing us the bound on generalization error in Section 5 using pseudodimension, and to Roland Freund and Tommi Jaakkola for useful comments on numerical methods.

References

Bartlett, P. L. (1998). The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, *44*(2), 525–536.

Bauer, E., & Kohavi, R. (to appear). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*.

Baum, E. B., & Haussler, D. (1989). What size net gives valid generalization?. *Neural Computation*, *1*(1), 151–160.

Blum, A. (1997). Empirical support for winnow and weighted-majority based algorithms: results on a calendar scheduling domain. *Machine Learning*, *26*, 5–23.

Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics*, *26*(3), 801–849.

Csiszár, I., & Tusnády, G. (1984). Information geometry and alternaning minimization procedures. *Statistics and Decisions, Supplement Issue*, *1*, 205–237.

Dietterich, T. G. (to appear). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*, 263–286.

Drucker, H., & Cortes, C. (1996). Boosting decision trees. In *Advances in Neural Information Processing Systems 8*, pp. 479–485.

Fletcher, R. (1987). *Practical Methods of Optimization* (Second edition). John Wiley.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conference*.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pp. 148–156.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Freund, Y., Schapire, R. E., Singer, Y., & Warmuth, M. K. (1997). Using and combining predictors that specialize. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pp. 334–343.

Friedman, J., Hastie, T., & Tibshirani, R. (1998). Additive logistic regression: a statistical view of boosting. Technical Report.

Haussler, D. (1992). Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, *100*(1), 78–150.

Haussler, D., & Long, P. M. (1995). A generalization of Sauer's lemma. *Journal of Combinatorial Theory, Series A*, *71*(2), 219–240.

Kearns, M., & Mansour, Y. (1996). On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*.

Maclin, R., & Opitz, D. (1997). An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 546–551.

Margineantu, D. D., & Dietterich, T. G. (1997). Pruning adaptive boosting. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pp. 211–218.

Merz, C. J., & Murphy, P. M. (1998). UCI repository of machine learning databases.. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Quinlan, J. R. (1996). Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 725–730.

Schapire, R. E. (1997). Using output codes to boost multiclass learning problems. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pp. 313–321.

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

Schapire, R. E., & Singer, Y. (to appear). BoosTexter: A boosting-based system for text categorization. *Machine Learning*.

Schwenk, H., & Bengio, Y. (1998). Training methods for adaptive boosting of neural networks. In *Advances in Neural Information Processing Systems 10*, pp. 647–653.

Appendix A

Properties of Z

In this appendix, we show that the function defined by Eq. (3) is a convex function in the parameters $a_1, \ldots, a_N$ and describe a numerical procedure based on Newton's method to find the parameters which minimize it.

To simplify notation, let $u_{ij} = -y_i g_j(x_i)$. We will analyze the following slightly more general form of Eq. (3)

$$\sum_{i=1}^{m} w_i \exp\left(\sum_{j=1}^{N} a_j u_{ij}\right) , \quad (w_i \geq 0, \sum_i w_i = 1) . \tag{A.1}$$

Note that in all cases discussed in this paper Z is of the form given by Eq. (A.1). We therefore refer for brevity to the function given by Eq. (A.1) as Z. The first and second order derivatives of Z with respect to $a_1, \ldots, a_N$ are

$$\nabla_k Z = \frac{\partial Z}{a_k} = \sum_{i=1}^{m} w_i \exp\left(\sum_{j=1}^{N} a_j u_{ij}\right) u_{ik} \tag{A.2}$$

$$\nabla_{kl}^2 Z = \frac{\partial^2 Z}{a_k a_l} = \sum_{i=1}^{m} w_i \exp\left(\sum_{j=1}^{N} a_j u_{ij}\right) u_{ik} u_{il} . \tag{A.3}$$

Denoting by $\mathbf{u}_i^T = (u_{i1}, \ldots, u_{iN})$ we can rewrite $\nabla^2 Z$ as

$$\nabla^2 Z = \sum_{i=1}^{m} w_i \exp\left(\sum_{j=1}^{N} a_j u_{ij}\right) \mathbf{u}\mathbf{u}^T .$$

Now, for any vector $\mathbf{x} \in \mathbb{R}^N$ we have that,

$$\begin{aligned}
\mathbf{x}^T \nabla^2 Z \, \mathbf{x} &= \mathbf{x}^T \left(\sum_{i=1}^{m} w_i \exp\left(\sum_{j=1}^{N} a_j u_{ij}\right) \mathbf{u}_i^T \mathbf{u}_i\right) \mathbf{x} \\
&= \sum_{i=1}^{m} w_i \exp\left(\sum_{j=1}^{N} a_j u_{ij}\right) \mathbf{x}^T \mathbf{u}_i \mathbf{u}_i^T \mathbf{x} \\
&= \sum_{i=1}^{m} w_i \exp\left(\sum_{j=1}^{N} a_j u_{ij}\right) (\mathbf{x} \cdot \mathbf{u}_i)^2 \geq 0 .
\end{aligned}$$

Hence, $\nabla^2 Z$ is positive semidefinite which implies that Z is convex with respect to $a_1, \ldots, a_N$ and has a unique minimum (with the exception of pathological cases).

To find the values of $a_1, \ldots, a_N$ that minimize Z we can use iterative methods such as Newton's method. In short, for Newton's method the new set of parameters is updated from the current set as follows

$$\mathbf{a} \leftarrow \mathbf{a} - \left(\nabla^2 Z\right)^{-1} \nabla Z^T , \tag{A.4}$$

where $\mathbf{a}^T = (a_1, \ldots, a_N)$.

Let

$$v_i = \frac{1}{Z} w_i \exp\left(\sum_{j=1}^{N} a_j u_{ij}\right) ,$$

and denote by

$$\mathrm{E}_{i\sim v}\left[\mathbf{u}_i\right] = \sum_{i=1}^{n} v_i \, \mathbf{u}_i \ \text{ and } \ \mathrm{E}_{i\sim v}\left[\mathbf{u}_i^T \mathbf{u}_i\right] = \sum_{i=1}^{n} v_i \, \mathbf{u}_i^T \mathbf{u}_i .$$

Then, substituting the values for ∇Z and $\nabla^2 Z$ from Eqs. (A.2) and (A.3) in Eq. (A.4), we get that the Newton parameter update is

$$\mathbf{a} \leftarrow \mathbf{a} - \left(\mathrm{E}_{i\sim v}\left[\mathbf{u}_i^T \mathbf{u}_i\right]\right)^{-1} \mathrm{E}_{i\sim v}\left[\mathbf{u}_i\right] .$$

Typically, the above update would result in a new set of parameters that attains a smaller value of Z than the current set. However, such a decrease is not *always* guaranteed. Hence, the above iteration should be augmented with a test on the value of Z and a line search in the direction of $\left(\nabla^2 Z\right)^{-1} \nabla Z^T$ in case of an increase in the value of Z. (For further details, see for instance Fletcher (1987)).

Appendix B

Bounding the Generalization Error

In this appendix, we prove a bound on the generalization error of the combined hypothesis produced by AdaBoost in terms of the margins of the training examples. An outline of the proof that we present here was communicated to us by Peter Bartlett. It uses techniques developed by Bartlett (1998) and Schapire et al. (1998).

Let $\mathcal{H}$ be a set of real-valued functions on domain $\mathcal{X}$. We let $\mathrm{co}(\mathcal{H})$ denote the *convex hull* of $\mathcal{H}$, namely,

$$\mathrm{co}(\mathcal{H}) = \left\{ f : x \mapsto \sum_h \alpha_h h(x) \mid \alpha_h \geq 0, \sum_h \alpha_h = 1 \right\}$$

where it is understood that each of the sums above are over the finite subset of hypotheses in $\mathcal{H}$ for which $\alpha_h > 0$. We assume here that the weights on the hypotheses are nonnegative. The result can be generalized to handle negative weights simply by adding to $\mathcal{H}$ all hypotheses $-h$ for $h \in \mathcal{H}$.

The main result of this appendix is the theorem below. This theorem is identical to Schapire et al.'s (1998) Theorem 2 except that we allow the weak hypotheses to be real-valued rather than binary.

We use $\mathrm{Pr}_{(x,y)\sim\mathcal{D}}[A]$ to denote the probability of the event A when the example (x, y) is chosen according to $\mathcal{D}$, and $\mathrm{Pr}_{(x,y)\sim S}[A]$ to denote probability with respect to choosing an example uniformly at random from the training set. When clear from context, we abbreviate these by $\mathrm{Pr}_{\mathcal{D}}[A]$ and $\mathrm{Pr}_S[A]$. We use $\mathrm{E}_{\mathcal{D}}[A]$ and $\mathrm{E}_S[A]$ to denote expected value in a similar manner.

To prove the theorem, we will first need to define the notion of a sloppy cover. For a class $\mathcal{F}$ of real-valued functions, a training set S of size m, and real numbers $\theta > 0$ and $\epsilon \geq 0$, we say that a function class $\hat{\mathcal{F}}$ is an *ϵ-sloppy θ-cover of $\mathcal{F}$ with respect to S* if, for all f in $\mathcal{F}$, there exists $\hat{f}$ in $\hat{\mathcal{F}}$ with $\mathrm{Pr}_{x\sim S}\left[|\hat{f}(x) - f(x)| > \theta\right] \leq \epsilon$. Let $\mathcal{N}(\mathcal{F}, \theta, \epsilon, m)$ denote the maximum, over all training sets S of size m, of the size of the smallest ϵ-sloppy θ-cover of $\mathcal{F}$ with respect to S.

THEOREM 8 *Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \{-1, +1\}$, and let S be a sample of m examples chosen independently at random according to $\mathcal{D}$. Suppose the weak-hypothesis space $\mathcal{H}$ of $[-1, +1]$-valued functions has pseudodimension d, and let $\delta > 0$. Assume that $m \geq d \geq 1$. Then with probability at least $1 - \delta$ over the random choice of the training set S, every weighted average function $f \in \mathrm{co}(\mathcal{H})$ satisfies the following generalization-error bound for all $\theta > 0$:*

$$\mathrm{Pr}_{\mathcal{D}}[yf(x) \leq 0] \leq \mathrm{Pr}_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\left(\frac{d \log^2(m/d)}{\theta^2} + \log(1/\delta)\right)^{1/2}\right).$$

Proof: Using techniques from Bartlett (1998), Schapire et al. (1998, Theorem 4) give a theorem which states that, for $\epsilon > 0$ and $\theta > 0$, the probability over the random choice of training set S that there exists any function $f \in \mathrm{co}(\mathcal{H})$ for which

$$\Pr_{\mathcal{D}}\left[yf(x) \leq 0\right] > \Pr_S\left[yf(x) \leq \theta\right] + \epsilon$$

is at most

$$2\mathcal{N}(\mathrm{co}(\mathcal{H}), \theta/2, \epsilon/8, 2m)e^{-\epsilon^2 m/32}. \tag{B.1}$$

We prove Theorem 8 by applying this result. To do so, we need to construct sloppy covers for $\mathrm{co}(\mathcal{H})$.

Haussler and Long (1995, Lemma 13) prove that

$$\mathcal{N}(\mathcal{H}, \theta, 0, m) \leq \sum_{i=0}^{d} \binom{m}{i} \left\lfloor \frac{1}{\theta} \right\rfloor^i \leq \left(\frac{em}{\theta d}\right)^d.$$

Fix any set $S \subseteq \mathcal{X}$ of size m. Then this result means that there exists $\hat{\mathcal{H}} \subseteq \mathcal{H}$ of cardinality $(em/(\theta d))^d$ such that for all $h \in \mathcal{H}$ there exists $\hat{h} \in \hat{\mathcal{H}}$ such that

$$\forall x \in S : |h(x) - \hat{h}(x)| \leq \theta. \tag{B.2}$$

Now let

$$\hat{\mathcal{C}}_N = \left\{ f : x \mapsto \frac{1}{N}\sum_{i=1}^{N} h_i(x) \mid h_i \in \hat{\mathcal{H}} \right\}$$

be the set of unweighted averages of N elements in $\hat{\mathcal{H}}$. We will show that $\hat{\mathcal{C}}_N$ is a sloppy cover of $\mathrm{co}(\mathcal{H})$.

Let $f \in \mathrm{co}(\mathcal{H})$. Then we can write

$$f(x) = \sum_j \alpha_j h_j(x)$$

where $\alpha_j \geq 0$ and $\sum_j \alpha_j = 1$. Let

$$\hat{f}(x) = \sum_j \alpha_j \hat{h}_j(x)$$

where $\hat{h}_j \in \hat{\mathcal{H}}$ is chosen so that h_j and $\hat{h}_j$ satisfy Eq. (B.2). Then for all $x \in S$,

$$\begin{aligned} |f(x) - \hat{f}(x)| &= \left| \sum_j \alpha_j (h_j(x) - \hat{h}_j(x)) \right| \\ &\leq \sum_j \alpha_j \left| h_j(x) - \hat{h}_j(x) \right| \\ &\leq \theta. \end{aligned} \tag{B.3}$$

Next, let us define a distribution $\mathcal{Q}$ over functions in $\hat{\mathcal{C}}_N$ in which a function $g \in \hat{\mathcal{C}}_N$ is selected by choosing $\hat{h}_1, \ldots, \hat{h}_N$ independently at random according to the distribution over $\hat{\mathcal{H}}$ defined by the α_j coefficients, and then setting $g = (1/N) \sum_{i=1}^{N} \hat{h}_i$. Note that, for fixed x, $\hat{f}(x) = \mathrm{E}_{g \sim \mathcal{Q}}[g(x)]$. We therefore can use Chernoff bounds to show that

$$\mathrm{Pr}_{g \sim \mathcal{Q}} \left[|\hat{f}(x) - g(x)| > \theta \right] \le 2e^{-N\theta^2/2}.$$

Thus,

$$\begin{aligned} &\mathrm{E}_{g \sim \mathcal{Q}} \left[\mathrm{Pr}_{(x,y) \sim S} \left[|\hat{f}(x) - g(x)| > \theta \right] \right] \\ &\quad = \mathrm{E}_{(x,y) \sim S} \left[\mathrm{Pr}_{g \sim \mathcal{Q}} \left[|\hat{f}(x) - g(x)| > \theta \right] \right] \le 2e^{-N\theta^2/2} . \end{aligned}$$

Therefore, there exists $g \in \hat{\mathcal{C}}_N$ such that

$$\mathrm{Pr}_{(x,y) \sim S} \left[|\hat{f}(x) - g(x)| > \theta \right] \le 2e^{-N\theta^2/2}.$$

Combined with Eq. (B.3), this means that $\hat{\mathcal{C}}_N$ is a $2e^{-N\theta^2/2}$-sloppy 2θ-cover of $\mathrm{co}(\mathcal{H})$. Since $|\hat{\mathcal{C}}_N| \le |\hat{\mathcal{H}}|^N$, we have thus shown that

$$\mathcal{N}(\mathrm{co}(\mathcal{H}), 2\theta, 2e^{-N\theta^2/2}, m) \le \left(\frac{em}{\theta d} \right)^{dN} .$$

Setting $N = (32/\theta^2) \ln(16/\epsilon)$, this implies that Eq. (B.1) is at most

$$2 \left(\frac{8em}{\theta d} \right)^{(32d/\theta^2)\ln(16/\epsilon)} e^{-\epsilon^2 m/32}. \tag{B.4}$$

Let

$$\epsilon = 16 \left(\frac{\ln(2/\delta)}{8m} + \frac{2d}{m\theta^2} \ln\left(\frac{8em}{d} \right) \ln\left(\frac{em}{d} \right) \right)^{1/2} . \tag{B.5}$$

Then the logarithm of Eq. (B.4) is

$$\begin{aligned} &\ln 2 - \frac{16d}{\theta^2} \ln\left(\frac{8em}{\theta d} \right) \ln\left(\frac{\ln(2/\delta)}{8m} + \frac{2d}{m\theta^2} \ln\left(\frac{8em}{d} \right) \ln\left(\frac{em}{d} \right) \right) \\ &- \ln(2/\delta) - \frac{16d}{\theta^2} \ln\left(\frac{8em}{d} \right) \ln\left(\frac{em}{d} \right) \\ &\le \ln\delta - \frac{16d}{\theta^2} \left(\ln\left(\frac{8em}{d} \right) \ln\left(\frac{em}{d} \right) - \ln\left(\frac{8em}{\theta d} \right) \ln\left(\frac{m\theta^2}{2d} \right) \right) \\ &\le \ln\delta. \end{aligned}$$

For the first inequality, we used the fact that $\ln(8em/d) \ge \ln(em/d) \ge 1$. For the second inequality, note that

$$\ln\left(\frac{8em}{\theta d} \right) \ln\left(\frac{m\theta^2}{2d} \right)$$

is increasing as a function of θ. Therefore, since $\theta \leq 1$, it is upper bounded by

$$\ln\left(\frac{8em}{d}\right)\ln\left(\frac{m}{2d}\right) \leq \ln\left(\frac{8em}{d}\right)\ln\left(\frac{em}{d}\right).$$

Thus, for the choice of ϵ given in Eq. (B.5), the bound in Eq. (B.4) is at most δ.

We have thus proved the bound of the theorem for a single given choice of $\theta > 0$ with high probability. We next prove that with high probability, the bound holds simultaneously for all $\theta > 0$. Let $\epsilon(\theta, \delta)$ be the choice of ϵ given in Eq. (B.5), regarding the other parameters as fixed. We have shown that, for all $\theta > 0$, the probability that

$$\mathrm{Pr}_{\mathcal{D}}\left[yf(x) \leq 0\right] > \mathrm{Pr}_{S}\left[yf(x) \leq \theta\right] + \epsilon(\theta, \delta) \tag{B.6}$$

is at most δ. Let $\Theta = \{1, 1/2, 1/4, \ldots\}$. By the union bound, this implies that, with probability at least $1 - \delta$,

$$\mathrm{Pr}_{\mathcal{D}}\left[yf(x) \leq 0\right] \leq \mathrm{Pr}_{S}\left[yf(x) \leq \theta\right] + \epsilon(\theta, \delta\theta/2) \tag{B.7}$$

for all $\theta \in \Theta$. This is because, for fixed $\theta \in \Theta$, Eq. (B.7) holds with probability $1 - \delta\theta/2$. Therefore, the probability that it fails to hold for *any* $\theta \in \Theta$ is at most $\sum_{\theta \in \Theta} \delta\theta/2 = \delta$.

Assume we are in the high probability case that Eq. (B.7) holds for all $\theta \in \Theta$. Then given any $\theta > 0$, choose $\theta' \in \Theta$ such that $\theta/2 \leq \theta' \leq \theta$. We have

$$\begin{aligned}
\mathrm{Pr}_{\mathcal{D}}\left[yf(x) \leq 0\right] &\leq \mathrm{Pr}_{S}\left[yf(x) \leq \theta'\right] + \epsilon(\theta', \delta\theta'/2) \\
&\leq \mathrm{Pr}_{S}\left[yf(x) \leq \theta\right] + \epsilon(\theta/2, \delta\theta/4).
\end{aligned}$$

Since

$$\epsilon(\theta/2, \delta\theta/4) = O\left(\frac{1}{\sqrt{m}}\left(\frac{d\log^2(m/d)}{\theta^2} + \log(1/\delta)\right)^{1/2}\right),$$

this completes the proof. ■