10 Multiclass Classification Problems

Until now, we have focused only on using boosting for binary classification problems in which the goal is to categorize each example into one of only two categories. In practice, however, we are often faced with *multiclass* problems in which the number of categories is more than two. For instance, in letter recognition, we need to categorize images into 26 categories for the letters $A, B, C, \ldots, Z$.

Although AdaBoost was designed explicitly for two-class problems, one might expect the generalization to the multiclass case to be natural and straightforward. In fact, there turn out to be many ways of extending boosting for multiclass learning, as will be seen in this chapter.

We begin with the most direct generalization of AdaBoost. This version, called AdaBoost.M1, has the advantage of simplicity and ease of implementation, but requires that the base classifiers be much better than random guessing, a condition that for many natural base learners cannot be fulfilled. For weaker base learners, this can be an insurmountable problem requiring a fundamentally different approach.

Aside from AdaBoost.M1, most multiclass versions of AdaBoost are based in some way on a reduction from the multiclass case to the simpler binary case. In other words, the problem of making a multi-way classification is replaced by multiple yes-no questions. For instance, if we are classifying images by the letter they represent, we might replace the question "What letter is this?" which has 26 possible answers, with 26 binary questions:

"Is it an A or not?"
"Is it a B or not?"
"Is it a C or not?"
$\vdots$

Clearly, if we can learn to answer these 26 questions accurately, then we can also answer the original classification question. This simple "one-against-all" approach is the basis of the multiclass AdaBoost.MH algorithm.

Asking so many questions might seem rather inefficient and, more importantly, one might notice that only one or two wrong answers to these binary questions are enough to cause an incorrect answer in the final classification. To alleviate this difficulty, we might consider asking more complex binary questions, such as:

"Is it a vowel or a consonant?"

"Is it in the first half of the alphabet?"

"Is it one of the letters in the word MACHINE?"

Given predicted answers to such binary questions for a particular instance, we can formulate a final classification by choosing the label that is "most consistent" with the binary responses. Even if many of these are erroneous, we still stand a reasonable chance of producing an overall prediction that is correct since the answers to the questions are more informative and overlap one another in the information they provide. Thus, such a scheme may be both more efficient and more robust.

Of course, the number of binary questions of this type grows extremely quickly, which means that the number of ways of reducing multiclass to binary is still more vast. Fortunately, as we will see, it turns out to be possible to study this problem in a general setting, and to derive and analyze an algorithm called AdaBoost.MO that can be applied to an entire family of reductions.

This general approach of reducing a more complicated learning problem to a simple binary classification problem can be applied in other situations as well. In particular, as we will see in chapter 11, we can use this method for ranking problems where the goal is to learn to rank a set of objects. For instance, we might want to rank documents by their relevance to a given search query. Once reduced to binary, an application of AdaBoost leads to a ranking algorithm called RankBoost. We can further view multiclass classification as a ranking problem, leading to yet another multiclass algorithm called AdaBoost.MR whose purpose is to rank the correct label higher than the incorrect ones.

Moreover, although we focus only on their combination with boosting, the reduction techniques that we present are quite general, and can certainly be applied to other learning methods as well, such as support-vector machines.

The ability of the algorithms in this chapter to generalize beyond the provided training set is clearly an issue of considerable importance. Nevertheless, we limit our scope in this chapter only to the study of performance on the training set. We note, however, that the techniques for analyzing generalization error given in previous chapters can certainly be applied to the multiclass setting as well (see exercises 10.3 and 10.4).

As an illustration, this chapter also includes an application of the presented techniques to the classification of caller utterances according to their meaning, a key component of spoken-dialogue systems.

10.1 A Direct Extension to the Multiclass Case

Our setup for multiclass learning is essentially the same as in the binary case, except that each label y_i is now assumed to belong to a set $\mathcal{Y}$ of all possible labels where the cardinality of $\mathcal{Y}$ may be greater than 2. For instance, in the letter recognition example above, $\mathcal{Y}$ would be the set $\{A, B, \ldots, Z\}$. Throughout this chapter, we denote this cardinality $|\mathcal{Y}|$ by K.

The first multiclass version of AdaBoost is called *AdaBoost.M1*—the *M* stands for multiclass, and the *1* distinguishes this extension as the first and most direct. As is natural, the weak learner in this setting generates hypotheses h which assign to each instance exactly one of the K possible labels so that $h : \mathcal{X} \to \mathcal{Y}$. Pseudocode for AdaBoost.M1 is shown as algorithm 10.1, and differs only slightly from binary AdaBoost (algorithm 1.1 (p. 5)). The goal of the weak learner is to generate on round t a base classifier h_t with low classification error

$$\epsilon_t \doteq \mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i],$$

just as for binary AdaBoost. The update to the distribution D_t is also the same as for AdaBoost, as given in the first form of the update from algorithm 1.1. The final hypothesis H is only slightly different: For a given instance x, H now outputs the label y that maximizes the sum of the weights of the weak hypotheses predicting that label. In other words, rather than computing a weighted majority vote as in binary classification, H computes the weighted *plurality* of the predictions of the base hypotheses.

In analyzing the training error of AdaBoost.M1, we require the same weak learning assumption as in the binary case, namely, that each weak hypothesis h_t have weighted error ϵ_t below $\frac{1}{2}$. When this condition is satisfied, theorem 10.1 (below) proves the same bound on the training error as for binary AdaBoost, showing that the error of the combined final hypothesis decreases exponentially, as in the binary case. This is of course good news for base learners that are able to meet this condition. Unfortunately, this requirement on the performance of the weak learner is much stronger than might be desired. In the binary case, when $K = 2$, a random guess will be correct with probability $\frac{1}{2}$, so the weak learning assumption posits performance only a little better than random. However, when $K > 2$, the probability of a correct random prediction is only $1/K$, which is less than $\frac{1}{2}$. Thus, our requirement that the accuracy of the weak hypothesis be greater than $\frac{1}{2}$ is significantly stronger than simply requiring that the weak hypothesis perform better than random guessing. For instance, with $K = 10$ classes, guessing randomly will give accuracy of 10%, which is far less than the 50% requirement.

Moreover, in the case of binary classification, a weak hypothesis h_t with error significantly *larger* than $\frac{1}{2}$ is of equal value to one with error significantly *less* than $\frac{1}{2}$ since h_t can be replaced by its negation $-h_t$ (an effect that happens "automatically" in AdaBoost, which chooses $\alpha_t < 0$ in such a case). However, for $K > 2$, a hypothesis h_t with error ϵ_t above

Algorithm 10.1
AdaBoost.M1: A first multiclass extension of AdaBoost

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

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \mathcal{Y}$.
- Aim: select h_t to minimalize the weighted error:

$$\epsilon_t \doteq \mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i] .$$

- If $\epsilon_t \geq \frac{1}{2}$, then set $T = t - 1$ and exit loop.
- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

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

Output the final hypothesis:

$$H(x) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \alpha_t \mathbf{1}\{h_t(x) = y\} .$$

$\frac{1}{2}$ is useless to the boosting algorithm, and generally cannot be converted into one with error below $\frac{1}{2}$. This is a significant difficulty. If such a weak hypothesis is returned by the weak learner, AdaBoost.M1, as we have presented it, simply halts, using only the weak hypotheses that were already computed (although there are other ways one might imagine for dealing with this).

In proving a bound on the training error, as in theorem 3.1, we give a slightly more general proof for the weighted training error with respect to an arbitrary initial distribution D_1.

Theorem 10.1 Given the notation of algorithm 10.1, assume that $\epsilon_t < \frac{1}{2}$ for all t, and let $\gamma_t \doteq \frac{1}{2} - \epsilon_t$. Let D_1 be an arbitrary initial distribution over the training set. Then the weighted training error of AdaBoost.M1's combined classifier H with respect to D_1 is bounded as

$$\mathbf{Pr}_{i\sim D_1}[H(x_i) \neq y_i] \leq \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

Proof The proof follows the same outline as in theorem 3.1, so we focus only on the differences from that proof.

First, let

$$F(x, y) \doteq \sum_{t=1}^{T} \alpha_t \left(\mathbf{1}\{y = h_t(x)\} - \mathbf{1}\{y \neq h_t(x)\}\right).$$

Noticing that

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t\left(\mathbf{1}\{y = h_t(x)\} - \mathbf{1}\{y \neq h_t(x)\}\right)\right)}{Z_t},$$

we can unravel this recurrence to obtain

$$D_{T+1}(i) = \frac{D_1(i)e^{-F(x_i, y_i)}}{\prod_{t=1}^{T} Z_t}.$$

If $H(x) \neq y$, then there exists a label $\ell \neq y$ such that

$$\sum_{t=1}^{T} \alpha_t \mathbf{1}\{y = h_t(x)\} \;\leq\; \sum_{t=1}^{T} \alpha_t \mathbf{1}\{\ell = h_t(x)\} \;\leq\; \sum_{t=1}^{T} \alpha_t \mathbf{1}\{y \neq h_t(x)\}.$$

The last inequality uses the fact that $\alpha_t \geq 0$ since $\epsilon_t < \frac{1}{2}$. Thus, $H(x) \neq y$ implies that $F(x, y) \leq 0$. Therefore, in general,

$$\mathbf{1}\{H(x) \neq y\} \leq e^{-F(x,y)}.$$

Now the same argument as in equation (3.5) gives that the (weighted) training error is

$$\sum_{i=1}^{m} D_1(i)\mathbf{1}\{H(x_i) \neq y_i\} \leq \prod_{t=1}^{T} Z_t.$$

Finally, the computation of Z_t beginning with equation (3.6) and leading to equation (3.9) is unchanged. ∎

It is disappointing, in the multiclass case, that we need to require such high accuracy, exceeding $\frac{1}{2}$, in order to analyze AdaBoost.M1. In fact, this difficulty turns out to be provably unavoidable when the performance of the weak learner is measured only in terms of error rate. This means that, in a sense, AdaBoost.M1 is the best we can hope for in a

multiclass boosting algorithm. To be more precise, we know that in the binary case, a base learner that performs slightly better than random on any distribution can always be used in conjunction with a boosting algorithm to achieve perfect training accuracy (and also arbitrarily good generalization accuracy, given sufficient data). Unfortunately, when $K > 2$, this is simply not possible, in general. We show this next with an example of a weak learner that consistently returns weak classifiers with accuracy significantly better than the random guessing rate of $1/K$, but for which no boosting algorithm can exist that uses such weak classifiers to compute a combined classifier with perfect (training) accuracy.

In this simple three-class example, we suppose that $\mathcal{X} = \{a, b, c\}$, $\mathcal{Y} = \{1, 2, 3\}$, and the training set consists of the three labeled examples $(a, 1)$, $(b, 2)$, and $(c, 3)$. Further, we suppose that we are using a base learner which chooses base classifiers that never distinguish between a and b. In particular, the base learner always chooses one of the following two base classifiers:

$$\hbar_1(x) = \begin{cases} 1 & \text{if } x = a \text{ or } x = b \\ 3 & \text{if } x = c, \end{cases}$$

or

$$\hbar_2(x) = \begin{cases} 2 & \text{if } x = a \text{ or } x = b \\ 3 & \text{if } x = c. \end{cases}$$

Then for any distribution over the training set, since a and b cannot both have weight exceeding $\frac{1}{2}$, it can be argued that either $\hbar_1$ or $\hbar_2$ will have accuracy at least $\frac{1}{2}$ (though not necessarily *exceeding* $\frac{1}{2}$, as would be necessary for theorem 10.1 to be of value here). This is substantially more than the accuracy of $\frac{1}{3}$ which would be achieved by pure random guessing among the three labels of $\mathcal{Y}$. However, regardless of how the training distributions are selected, and regardless of how the collected base classifiers are combined, a final classifier H that bases its predictions only on those of the base hypotheses will necessarily classify a and b in exactly the same way, and therefore will misclassify at least one of them. Thus, the training accuracy of H on the three examples can never exceed $\frac{2}{3}$, so perfect accuracy cannot be achieved by any boosting method. (This argument is somewhat informal in its treatment of the notion of a general boosting algorithm; a more rigorous proof could be devised along the lines of the lower bound proved later in section 13.2.2. See exercise 13.10.)

Despite this limitation, in practice AdaBoost.M1 works quite effectively when using fairly strong base classifiers, such as decision trees and neural networks, which typically are able to find base classifiers with accuracy surpassing $\frac{1}{2}$, even on the difficult distributions constructed by boosting. For instance, in section 1.2.2, we looked at how AdaBoost performs on a range of benchmark datasets using C4.5, the decision-tree learning algorithm, as a base learner. Eleven of the datasets that were used were multiclass, ranging from

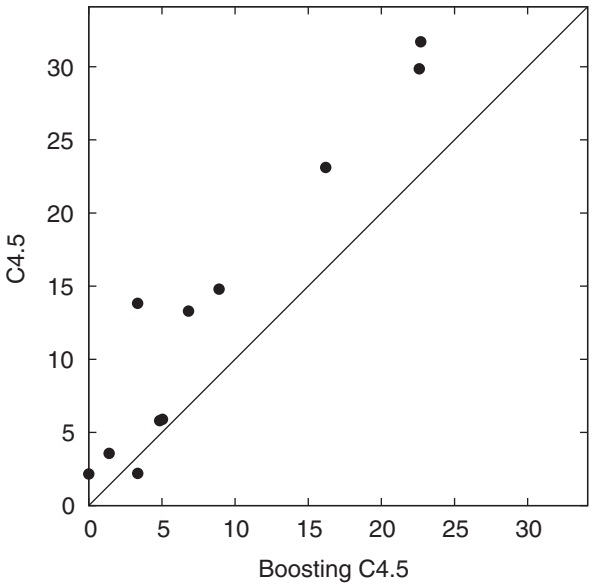


Figure 10.1
A comparison of C4.5 with and without boosting on 11 multiclass benchmark datasets. See figure 1.3 (p. 12) for further explanation.

3 up to 26 classes. Figure 10.1 shows the subset of the results in figure 1.3 that involve these 11 multiclass datasets. From these experiments, we see that AdaBoost.M1, when used in combination with C4.5 (which itself handles multiclass problems directly), gives performance that is strong overall, and apparently is not adversely affected by the 50%-accuracy requirement.

On the other hand, when using the much weaker decision stumps as base hypotheses (which also can be modified directly for multiclass problems), AdaBoost.M1 is not even able to get off the ground on the nine datasets with four or more classes—on each one of these, the best decision stump found on the very first round of boosting already has an error exceeding 50% (which is why AdaBoost.M1 was not used in the experiments reported in section 1.2.2). For the two datasets with exactly three classes, AdaBoost.M1 successfully improves the test error, from 35.2% to 4.7% in one case, and from 37.0% to 9.2% in the other; however, in the second case, a different multiclass method does significantly better, achieving a test error of 4.4%.

Thus, when using weak base classifiers, AdaBoost.M1 is inadequate for multiclass problems. In the remainder of this chapter, as well as in section 11.4, we develop multiclass boosting techniques that place far less stringent demands on the weak learning algorithm (including the method that was actually used in the decision-stump experiments reported in section 1.2.2).

10.2 The One-against-All Reduction and Multi-label Classification

To make boosting possible even with a very weak base learner, in one way or another, the communication between the boosting algorithm and the base learner must be augmented. We will shortly see several examples of how this can be done. As discussed earlier, the multiclass algorithms that we now focus on are generally based on reductions to the binary case. The first of these, called AdaBoost.MH, is based on a "one-against-all" reduction in which the multi-way classification problem is reduced to several binary problems, each asking if a given instance is or is not an example of a particular class. Each of these binary problems could be treated independently, training a separate copy of AdaBoost for each. Here, instead, we pursue a more unified approach in which the binary problems are all handled together and simultaneously in a single run of the boosting algorithm.

Furthermore, in this section, we consider the more general *multi-label* setting in which each example may be assigned 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 general topic; for example, an article about a presidential candidate who throws out the first ball at a major-league baseball game should be classified as belonging to both the "politics" and the "sports" categories.

10.2.1 Multi-label Classification

As before, $\mathcal{Y}$ is a finite set of labels or classes of cardinality K. In the standard single-label classification setting considered up to this point, each example $x \in \mathcal{X}$ is assigned a single class $y \in \mathcal{Y}$ so that labeled examples are pairs (x, y). The goal then, typically, is to find a hypothesis $H : \mathcal{X} \to \mathcal{Y}$ that minimizes the probability that $y \neq H(x)$ on a newly observed example (x, y). In contrast, 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 that 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 $\text{one-err}_D(H)$. That is,

$$\text{one-err}_D(H) \doteq \mathbf{Pr}_{(x,Y)\sim D}[H(x) \notin Y] .$$

Note that for single-label classification problems, the one-error is identical to ordinary classification error. In what follows, and in chapter 11, we introduce other loss measures that

can be used in the multi-label setting. We begin with one of these called the Hamming loss, and we show how its minimization also leads to an algorithm for the standard single-label multiclass case.

10.2.2 Hamming Loss

Rather than predicting just one label correctly, the goal might instead be to predict all and only all of the correct labels. In this case, the learning algorithm generates a hypothesis that 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

$$\frac{1}{K} \cdot \mathbf{E}_{(x,Y)\sim D}[\ |H(x) \triangle Y|\ ] \tag{10.1}$$

where $A \triangle B$ denotes the *symmetric difference* between two sets A and B, that is, the set of elements in exactly one of the two sets. (The $1/K$ appearing in equation (10.1) is meant merely to ensure a value in $[0, 1]$.) We call this measure the *Hamming loss* of H, and we denote it by $\text{hloss}_D(H)$.

When D is the empirical distribution (that is, the uniform distribution over the m training examples), we denote this empirical Hamming loss by $\widehat{\text{hloss}}(H)$. Similarly, empirical one-error is denoted $\widehat{\text{one-err}}(H)$.

To minimize Hamming loss, we can, in a natural way, decompose the problem into K orthogonal binary classification problems following the intuitive one-against-all approach that was described earlier. That is, we can think of Y as specifying K binary labels, each depending on whether some 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 indicate ℓ's inclusion in Y:

$$Y[\ell] \doteq \begin{cases} +1 & \text{if } \ell \in Y \\ -1 & \text{if } \ell \notin Y. \end{cases} \tag{10.2}$$

Thus, we can identify any subset Y with a vector in $\{-1, +1\}^K$ or, equivalently, a function mapping $\mathcal{Y}$ to $\{-1, +1\}$. Throughout this chapter, we will move fluidly between these two equivalent representations of Y either as a subset or as a binary vector/function. 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

$$H(x, \ell) \doteq H(x)[\ell] = \begin{cases} +1 & \text{if } \ell \in H(x) \\ -1 & \text{if } \ell \notin H(x). \end{cases}$$

The Hamming loss given in equation (10.1) can then be rewritten as

Algorithm 10.2
AdaBoost.MH: A multiclass, multi-label version of AdaBoost based on Hamming loss

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 $i = 1, \ldots, m$ and $\ell \in \mathcal{Y}$ (where $K = |\mathcal{Y}|$).
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 $\alpha_t \in \mathbb{R}$.
- Aim: select h_t and α_t to minimalize the normalization factor

$$Z_t \doteq \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} D_t(i, \ell) \exp(-\alpha_t \, Y_i[\ell] \, h_t(x_i, \ell)).$$

- Update, for $i = 1, \ldots, m$ and for $\ell \in \mathcal{Y}$:

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

Output the final hypothesis:

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

$$\frac{1}{K} \sum_{\ell \in \mathcal{Y}} \mathbf{Pr}_{(x,Y)\sim D}[\, H(x, \ell) \neq Y[\ell] \,].$$

(Technically, we sometimes allow H to output a prediction of 0 which, in this definition, is always counted as an error.)

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 $\ell \in \mathcal{Y}$. In other words, each instance is actually a pair of the form (x_i, ℓ) whose binary label is $+1$ if $\ell \in Y_i$, and -1 otherwise. The result is the boosting algorithm called *AdaBoost.MH—M* for multiclass, *H* for Hamming. As shown in algorithm 10.2, the procedure maintains a distribution D_t over examples i and labels ℓ. On round t, the weak learner accepts the distribution D_t (as well as the training set), and generates a weak hypothesis $h_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. In this way, the

communication between the booster and the weak learner is significantly richer than in AdaBoost.M1, where both D_t and h_t were of a much simpler form. We can interpret $h_t(x, \ell)$ as a confidence-rated prediction of whether label ℓ should or should not be assigned to example x, as indicated by the sign of the prediction (with the magnitude measuring confidence). Our reduction also leads to the choice of final hypothesis shown in the algorithm.

Note that we have adopted a general approach that admits the use of confidence-rated predictions as in chapter 9. As such, we have left α_t unspecified. Continuing with this approach, we can see that the analysis for the binary case given by theorem 9.1 can be combined with the reduction used to derive this algorithm, yielding the following bound on the Hamming loss of the final hypothesis:

Theorem 10.2 Assuming the notation of algorithm 10.2, the empirical Hamming loss of AdaBoost.MH's final hypothesis H is at most

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

We can immediately adapt ideas from chapter 9 to this binary classification problem. As before, theorem 10.2 suggests that our goal, in the choice of both h_t and α_t, should be to minimalize

$$Z_t \doteq \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} D_t(i, \ell) \exp(-\alpha_t Y_i[\ell] h_t(x_i, \ell)) \tag{10.3}$$

on each round. For instance, if we require that each h_t have range $\{-1, +1\}$, then we should choose

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{10.4}$$

where

$$\epsilon_t \doteq \mathbf{Pr}_{(i,\ell) \sim D_t}[h_t(x_i, \ell) \neq Y_i[\ell]]$$

can be thought of as a weighted Hamming loss with respect to D_t. As before, this choice gives

$$Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)}.$$

So, to minimize Z_t, the weak learning algorithm should choose ϵ_t as far from $\frac{1}{2}$ as possible; in other words, it should seek to minimize the Hamming loss weighted by D_t. Note that, as in the binary case, if the base classifier guesses randomly, then ϵ_t will be equal to $\frac{1}{2}$; moreover,

any value of ϵ_t that is bounded away from $\frac{1}{2}$ will give Z_t that is strictly less than 1, thus ensuring eventual perfect training accuracy. Thus, whereas AdaBoost.M1 required a weak learning algorithm that must be very substantially better than random, we see that AdaBoost.MH can be used with any weak learner that is just slightly better than random guessing.

We also can combine these ideas with those in section 9.2.6 on domain-partitioning weak hypotheses. As in that section, suppose that h is associated with a partition $X_1, \ldots, X_J$ of the space $\mathcal{X}$. It is natural then to create a partition of the set $\mathcal{X} \times \mathcal{Y}$ consisting of all sets $X_j \times \{\ell\}$ for $j = 1, \ldots, J$ and $\ell \in \mathcal{Y}$. An appropriate hypothesis h can then be formed which predicts $h(x, \ell) = c_{j\ell}$ for $x \in X_j$. Applied to the current setting, equation (9.15) implies that we should choose

$$c_{j\ell} = \frac{1}{2} \ln \left(\frac{W_+^{j\ell}}{W_-^{j\ell}} \right) \tag{10.5}$$

where

$$W_b^{j\ell} \doteq \sum_{i=1}^{m} D(i, \ell) \mathbf{1}\left[x_i \in X_j \wedge Y_i[\ell] = b \right].$$

(In this case, as in section 9.2.6, the α_t's are fixed to be 1.) By equation (9.16), this choice of $c_{j\ell}$ gives

$$Z_t = 2 \sum_{j=1}^{J} \sum_{\ell \in \mathcal{Y}} \sqrt{W_+^{j\ell} W_-^{j\ell}}. \tag{10.6}$$

So, in a manner similar to that described in section 9.2.6 for the binary case, we can design base learners that seek domain-partitioning base classifiers, such as (multi-label) decision stumps, based on the criterion in equation (10.6), and giving real-valued predictions as in equation (10.5).

10.2.3 Relation to One-Error and Single-Label Classification

We can use AdaBoost.MH even when the goal is to minimize one-error. Perhaps the most natural way to do this is to define a classifier H^1 that predicts the label y for which the weighted sum of the weak-hypothesis predictions in favor of y is greatest; that is,

$$H^1(x) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \alpha_t h_t(x, y). \tag{10.7}$$

The next simple theorem relates the one-error of H^1 and the Hamming loss of H.

Theorem 10.3 With respect to any distribution D over observations (x, Y) with $\emptyset \neq Y \subseteq \mathcal{Y}$,

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

(where $K = |\mathcal{Y}|$).

Proof Assume $Y \neq \emptyset$ and suppose $H^1(x) \notin Y$. We argue that this implies that $H(x, \ell) \neq Y[\ell]$ for some $\ell \in \mathcal{Y}$. First, suppose the maximum in equation (10.7) is strictly positive, and let $\ell = H^1(x)$ realize the maximum. Then $H(x, \ell) = +1$ (since the maximum is positive), but $Y[\ell] = -1$ since $\ell \notin Y$. On the other hand, if the maximum in equation (10.7) is non-positive, then $H(x, \ell)$ is 0 or -1 for all $\ell \in \mathcal{Y}$, but $Y[\ell] = +1$ for some $\ell \in \mathcal{Y}$ since Y is not empty.

Thus, in either case, if $H^1(x) \notin Y$, then $H(x, \ell) \neq Y[\ell]$ for some $\ell \in \mathcal{Y}$. This implies that

$$\mathbf{1}\{H^1(x) \notin Y\} \leq \sum_{\ell \in \mathcal{Y}} \mathbf{1}\{H(x, \ell) \neq Y[\ell]\},$$

which, taking expectations of both sides with respect to $(x, Y) \sim D$, yields the theorem. ∎

In particular, this means that AdaBoost.MH can be applied to single-label multiclass classification problems. Combining theorems 10.2 and 10.3 results in a bound on the training error of the final hypothesis H^1 that is at most

$$K \prod_{t=1}^{T} Z_t \tag{10.8}$$

where Z_t is as in equation (10.3). In fact, theorem 10.4 below will imply a better bound of

$$\frac{K}{2} \prod_{t=1}^{T} Z_t \tag{10.9}$$

for the one-error of AdaBoost.MH when applied to single-label problems. 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 0-th weak hypothesis is chosen that predicts -1 on all example-label pairs; that is, $h_0 \equiv -1$. For this weak hypothesis, $\epsilon_0 = 1/K$, 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 equation (10.9), 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 algorithm 10.2 only in the manner in which D_1 is initialized. Specifically, D_1 should be chosen so that

$$D_1(i, \ell) = \begin{cases} 1/(2m) & \text{if } \ell = y_i \\ 1/[2m(K-1)] & \text{else.} \end{cases}$$

Note that H^1 is unaffected.

An alternative to the approach taken in equation (10.7) would choose any label y for which $H(x, y) = +1$. This approach has the advantage of being less specific to the representation used by the learning algorithm but, on the other hand, fails to take into account the strength of the predictions for each class, which we expect to be highly informative. It will be possible to obtain an analysis of this alternate approach for single-label problems as a special case of theorem 10.4.

10.3 Application to Semantic Classification

As a typical example of how these ideas can be applied, consider the problem of categorizing the type of call requested by a phone customer of the telecommunications company AT&T. Some examples of spoken customer utterances and their correct classifications are shown in table 10.2. In this problem, there are 15 predefined categories, shown in table 10.1, intended to capture the caller's intention. Most of these are requests for information or specific services, or instructions on how a call is to be billed. Note that this is in fact a multi-label problem—the same utterance may have multiple labels.

To apply boosting, we can use AdaBoost.MH since it is designed for such multiclass, multi-label problems. We next need to select or design a base learning algorithm. Here, we choose the very simple decision stumps mentioned above. Specifically, each such classifier first tests a given document for the presence or absence of a particular term. A "term" can be a single word (such as *collect*), a pair of adjacent words (such as *my home*), or a possibly sparse triple of adjacent words (such as *person ? person*, which will match any word in place of the question mark, such as in the phrase "person to person"). The presence or absence of a term partitions the domain of all possible documents into two disjoint sets, so

Table 10.1
The classes in the call classification task

AC	AreaCode	CM	Competitor	RA	Rate
AS	AttService	DM	DialForMe	3N	ThirdNumber
BC	BillingCredit	DI	Directory	TI	Time
CC	CallingCard	HO	HowToDial	TC	TimeCharge
CO	Collect	PP	PersonToPerson	OT	Other

Table 10.2
Some typical example utterances and their classifications in the call classification task

yes I'd like to place a collect call long distance please	Collect
operator I need to make a call but I need to bill it to my office	ThirdNumber
yes I'd like to place a call on my master card please	CallingCard
I'm trying to make a calling card call to five five five one two one two in chicago	CallingCard, DialForMe
I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill	BillingCredit
yeah I need to make a collect call to bob	Collect, PersonToPerson

we can apply the techniques of section 10.2.2 to define a criterion for selecting the "best" base classifier on each round (equation (10.6)), and also for selecting a set of values for every label which will be output by the base classifier, depending on whether or not the term is present.

This leads to base classifiers of the form given in figures 10.2 and 10.3, which show the first several base classifiers found by boosting on the actual dataset. For instance, the second one says roughly in words:

If the word *card* appears in what was said, then predict positively for the class CallingCard with high confidence, and negatively (with varying degrees of confidence) for each of the other classes; otherwise, if *card* does not appear, then predict negatively for the CallingCard class, and abstain on each of the other classes.

Many of the terms found seem natural for this task, such as *collect*, *card*, *my home*, and *person ? person*. This suggests boosting's usefulness for selecting "features" from a very large space of candidates. It is curious, however, that on many rounds, terms that seem unimportant are chosen, such as *I*, *how*, and *and*. In such cases, it may be that these words are more useful than might be guessed, perhaps because they are used or not used in typical phrasings of these sorts of requests. In the case of *and* on round 13, we see in fact that all of the predictions are low confidence, suggesting that this term, although selected, is rather unimportant.

As can be seen on this dataset, boosting can also be used as a method for identifying outliers. This is because such mislabeled or highly ambiguous examples tend to receive the most weight under the distributions computed by boosting. For example, table 10.3 is a list of some of the examples with the highest weight under the final distribution computed by boosting. Most of these examples are indeed outliers, many of which are clearly mislabeled. In practice, once identified, such examples could either be removed entirely from the dataset, or their labels corrected by hand.

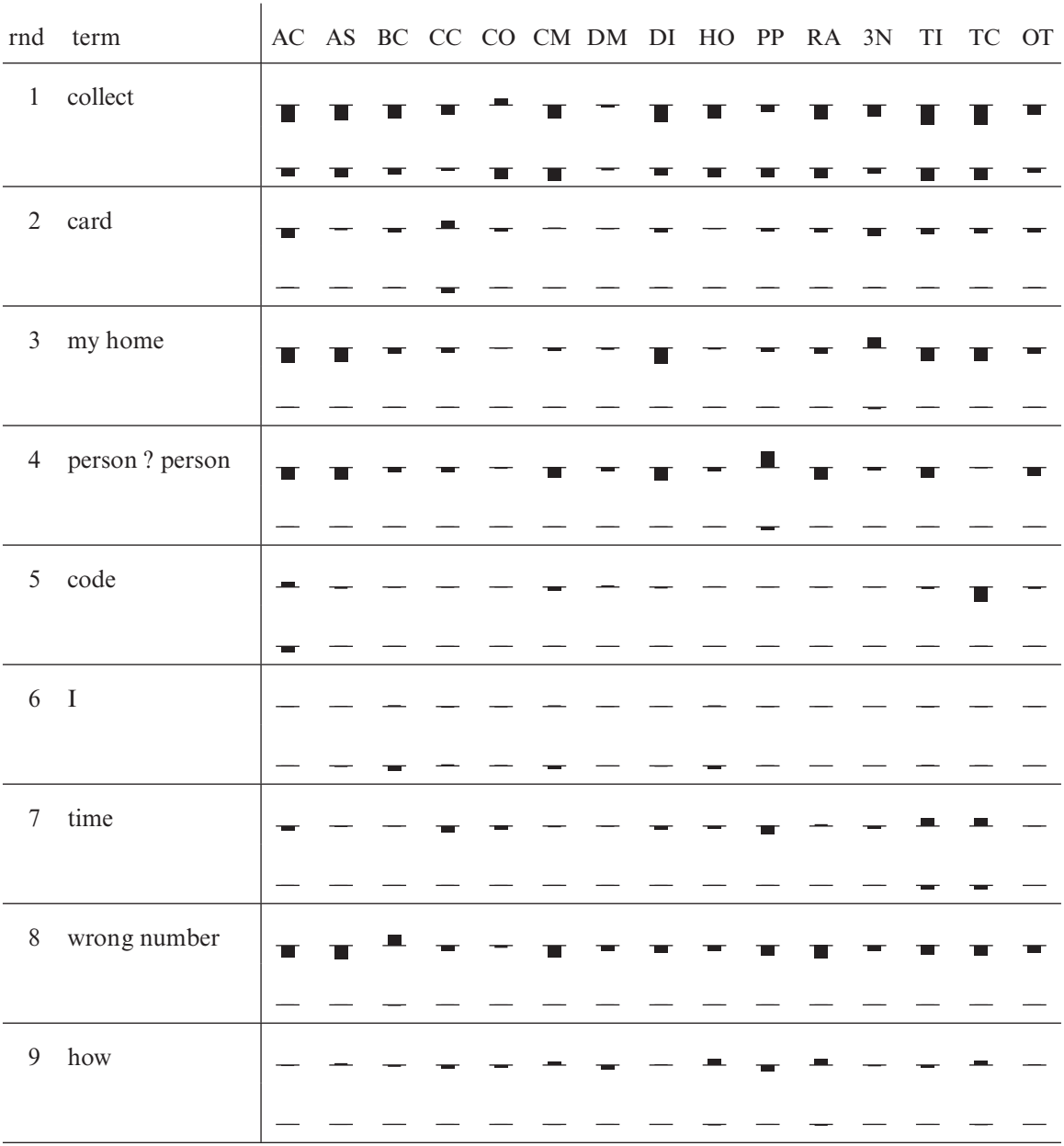


Figure 10.2
The first nine weak hypotheses found when confidence-rated AdaBoost.MH is run on the call classification task using the weak learning algorithm described in the text. Each weak hypothesis has the following form and interpretation: If the term associated with the weak hypothesis occurs in the given document, then output the first row of values; otherwise, output the second row of values. Here, each value, represented graphically as a bar, gives the output of the weak hypothesis for one of the classes, which may be positive or negative.

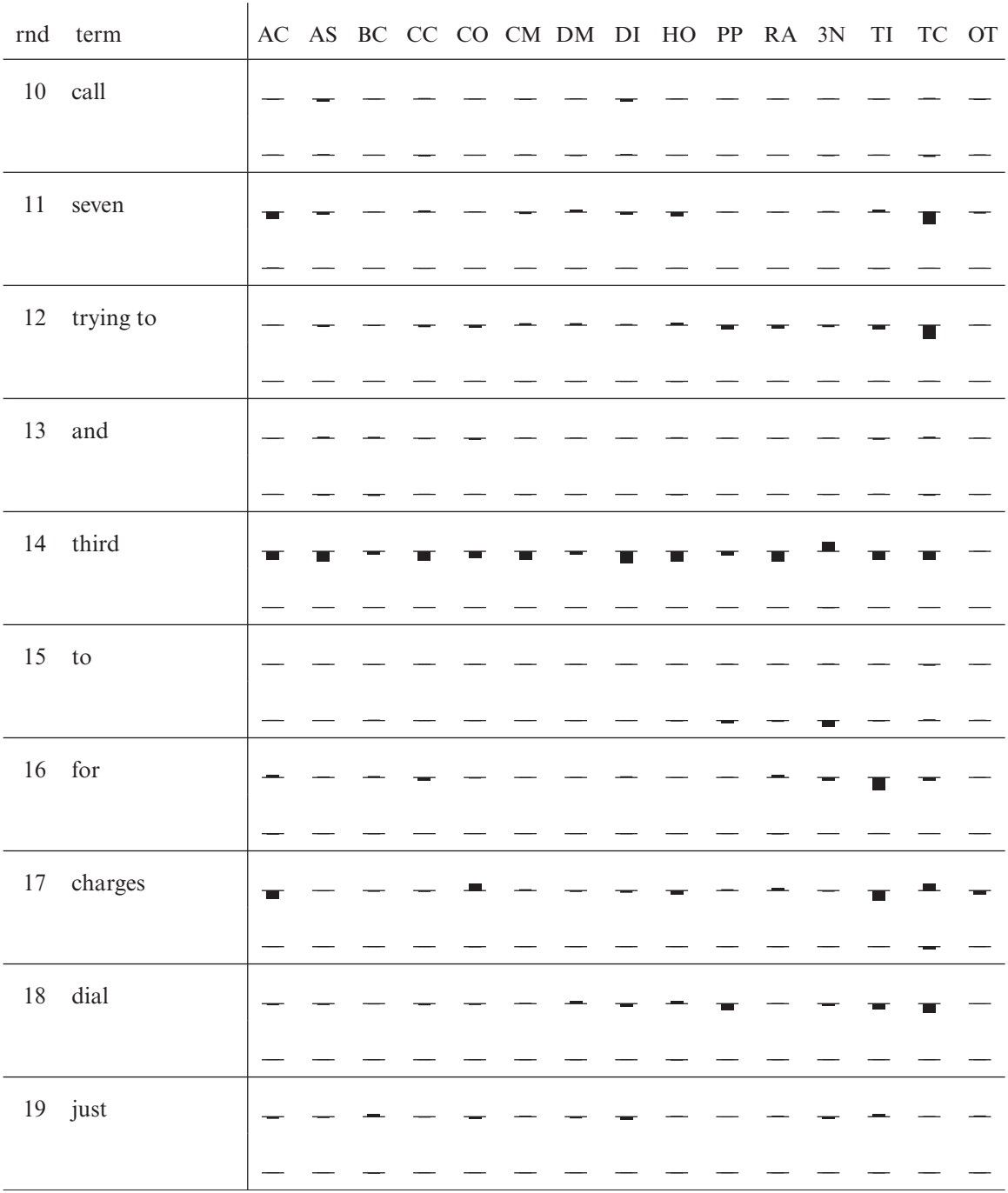


Figure 10.3
The next ten weak hypotheses (continuing figure 10.2).

Table 10.3
Examples with the highest final weight on the call classification task

I'm trying to make a credit card call	Collect
hello	Rate
yes I'd like to make a long distance collect call please	CallingCard
calling card please	Collect
yeah I'd like to use my calling card number	Collect
can I get a collect call	CallingCard
yes I would like to make a long distant telephone call and have the charges billed to another number	CallingCard, DialForMe
yeah I can not stand it this morning I did oversea call is so bad	BillingCredit
yeah special offers going on for long distance	AttService, Rate
mister xxxxx please william xxxxx	PersonToPerson
yes ma'am I I'm trying to make a long distance call to a non dialable point in san miguel philippines	AttService, Other
yes I like to make a long distance call and charge it to my home phone that's where I'm calling at my home	DialForMe

Many of the labels supplied by human annotators are obviously incorrect.

10.4 General Reductions Using Output Codes

As described in section 10.2, AdaBoost.MH solves multiclass single-label classification problems by reducing to a set of binary problems using a very straightforward one-against-all approach. In this section, we describe a much more general technique that encompasses a large family of reductions from multiclass to binary.

10.4.1 Multiclass to Multi-label

In fact, we can think about these reductions formally as mappings from the given single-label problem to a multi-label formulation. The method used in section 10.2 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 $(x, \{y\})$. In other words, an example with label y is mapped to a multi-label example with a label set consisting of the singleton $\{y\}$, meaning that y is the one and only label that should be associated with this instance. When combined with AdaBoost.MH, the result is a multiclass boosting algorithm based on this one-against-all reduction.

However, it is generally possible, and often desirable, to use a more sophisticated mapping corresponding to some other multiclass-to-binary reduction. In general, we can use any injective (one-to-one) mapping $\Omega : \mathcal{Y} \to 2^{\overline{\mathcal{Y}}}$ for this purpose. This mapping specifies how each example should be relabeled or "coded" to create a new multi-label example. In particular, each example (x, y) gets mapped to $(x, \Omega(y))$; that is, every label y is replaced by multi-label set $\Omega(y)$. Note that Ω maps to subsets of an unspecified label set $\overline{\mathcal{Y}}$ of cardinality

$\overline{K} = |\overline{\mathcal{Y}}|$ which typically is not the same as $\mathcal{Y}$. Intuitively, each element $\overline{y}$ of $\overline{\mathcal{Y}}$ is a binary question or dichotomy. Examples (x, y) (in the original problem) for which $\overline{y} \in \Omega(y)$ are positive examples for class $\overline{y}$, while those for which $\overline{y} \notin \Omega(y)$ are the negative examples of $\overline{y}$. Equivalently, we can identify Ω with a $K \times \overline{K}$ $\{-1, +1\}$-valued *coding matrix* where

$$\Omega(y, \overline{y}) \doteq \Omega(y)[\overline{y}] = \begin{cases} +1 & \text{if } \overline{y} \in \Omega(y) \\ -1 & \text{else,} \end{cases}$$

and where we continue to identify $\Omega(y)$, a subset, with a vector in $\{-1, +1\}^{\overline{K}}$. Each $\overline{y} \in \overline{\mathcal{Y}}$ corresponds, then, to a binary question in which each example (x, y) (in the original problem) is given the binary label $\Omega(y, \overline{y})$.

For example, the reduction of section 10.2 is obtained simply by setting $\overline{\mathcal{Y}} = \mathcal{Y}$ and $\Omega(y) = \{y\}$ for all y. A more interesting example is given in table 10.4. At the top is a sample coding matrix Ω mapping the original label set $\mathcal{Y} = \{a, b, c, d\}$ to the mapped label set $\overline{\mathcal{Y}} = \{1, 2, 3\}$. According to this coding, label 1 in $\overline{\mathcal{Y}}$ asks if an example's label is in $\{a, d\}$ or if it is in the complement $\{b, c\}$; label 2 asks if it is in $\{a, b\}$ or $\{c, d\}$; and label 3 asks if it is in $\{b, c, d\}$ or $\{a\}$. So columns of this matrix can be viewed as binary problems or dichotomies between one set of labels and another. The rows, on the other hand, can be viewed as binary "codewords" that encode the original set of labels: a as $\langle +1, +1, -1\rangle$, b as $\langle -1, +1, +1\rangle$, and so on. The bottom of the figure shows how a small dataset would be relabeled by Ω. We can regard the mapped labels either as multi-label sets or as the

Table 10.4
A sample coding matrix (top) and its effect on a sample dataset (bottom)

Ω	1	2	3
a	$+1$	$+1$	-1
b	-1	$+1$	$+1$
c	-1	-1	$+1$
d	$+1$	-1	$+1$

	Dichotomies			
original	1	2	3	multi-label
$(x_1, a) \rightarrow$	$(x_1, +1)$	$(x_1, +1)$	$(x_1, -1)$	$= (x_1, \{1, 2\})$
$(x_2, c) \rightarrow$	$(x_2, -1)$	$(x_2, -1)$	$(x_2, +1)$	$= (x_2, \{3\})$
$(x_3, a) \rightarrow$	$(x_3, +1)$	$(x_3, +1)$	$(x_3, -1)$	$= (x_3, \{1, 2\})$
$(x_4, d) \rightarrow$	$(x_4, +1)$	$(x_4, -1)$	$(x_4, +1)$	$= (x_4, \{1, 3\})$
$(x_5, b) \rightarrow$	$(x_5, -1)$	$(x_5, +1)$	$(x_5, +1)$	$= (x_5, \{2, 3\})$

A multiclass, single-label example in the original dataset (on the left of the bottom table) gets mapped to three binary examples, one for each of the three dichotomies associated with this code (middle columns) or, equivalently, to a single, multi-label example (right column).

binary labels of the three dichotomies defined by the columns of Ω. For instance, (x_3, a) gets mapped to the multi-labeled example $(x_3, \{1, 2\})$ or, equivalently, it can be viewed as a positive example for dichotomies 1 and 2, and a negative example for dichotomy 3.

Once such a multi-label formulation Ω has been chosen, we can apply AdaBoost.MH directly to the transformed, multi-labeled training set. The result is a learning method that attempts to solve all of the binary problems associated with Ω simultaneously. Because this mapping is arbitrary, the technique and analysis that we now describe are quite general, and can be applied to any multiclass-to-binary reduction.

After applying AdaBoost.MH to the transformed data, how should we classify a new instance x? The most direct idea is to evaluate AdaBoost.MH's final classifier H on x, and then to choose the label $y \in \mathcal{Y}$ for which the mapped codeword $\Omega(y)$ is closest in Hamming distance (that is, the number of coordinates where two binary vectors disagree) to $H(x, \cdot)$. In other words, we predict the label y which minimizes

$$\sum_{\overline{y} \in \overline{\mathcal{Y}}} \mathbf{1}\{\Omega(y, \overline{y}) \neq H(x, \overline{y})\}.$$

This is called *Hamming decoding*.

A weakness of this approach is that it ignores the confidence with which each label was included or not included in the label set predicted by H. 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 total weight under the final distribution for the reduction induced by Ω, and thus most closely fits the learned combined hypothesis. In other words, the idea is to predict the label y which minimizes

$$\sum_{\overline{y} \in \overline{\mathcal{Y}}} \exp(-\Omega(y, \overline{y})\ F(x, \overline{y}))$$

where $F(x, \overline{y}) \doteq \sum_{t=1}^{T} \alpha_t h_t(x, \overline{y})$ is the weighted sum of weak hypotheses output by AdaBoost.MH. This expression also represents the exponential loss (see section 7.1) associated with example (x, y) for this reduction. We therefore call this approach *loss-based decoding*. The resulting algorithm is called *AdaBoost.MO—M* for multiclass, *O* for output coding. Pseudocode is given as algorithm 10.3, including both Hamming and loss-based decoding variants.

How should we choose the code Ω? The one-against-all reduction corresponds to a square matrix with $+1$ on the diagonal and -1 in all other entries, as shown at the top of table 10.5 for a four-class problem. Intuitively, however, it is often desirable to map different labels to sets or codewords which are far from one another, say, in terms of their symmetric difference, or Hamming distance. Such a reduction will be richly redundant, and thus robust in the information that each binary problem provides. The idea is that if all of the codewords are far apart, then even if $H(x, \cdot)$ is incorrect in its predictions on many of the mapped labels, the codeword corresponding to the correct label will remain the closest, so that the overall prediction will still be correct.

Algorithm 10.3
AdaBoost.MO: A multiclass version of AdaBoost based on output codes

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$
output code $\Omega : \mathcal{Y} \to 2^{\overline{\mathcal{Y}}}$.

- Run AdaBoost.MH on relabeled data: $(x_1, \Omega(y_1)), \ldots, (x_m, \Omega(y_m))$.
- Get back final hypothesis H of form $H(x, \overline{y}) = \text{sign}(F(x, \overline{y}))$
  where $F(x, \overline{y}) \doteq \sum_{t=1}^{T} \alpha_t h_t(x, \overline{y})$.
- Output modified final hypothesis:

$$H^{ham}(x) = \arg\min_{y \in \mathcal{Y}} \sum_{\overline{y} \in \overline{\mathcal{Y}}} \mathbf{1}\{\Omega(y, \overline{y}) \neq H(x, \overline{y})\} \quad \text{(Hamming decoding)}$$

or

$$H^{lb}(x) = \arg\min_{y \in \mathcal{Y}} \sum_{\overline{y} \in \overline{\mathcal{Y}}} \exp\left(-\Omega(y, \overline{y})\, F(x, \overline{y})\right) \quad \text{(loss-based decoding).}$$

This is the essence of an approach known as *error-correcting output coding* which uses codes that have been designed with exactly such an error-correcting property. Note that when $\overline{K}$ is not too small, even an entirely random code Ω is likely to have this property. Alternatively, when K is not too large, we can use a *complete* code consisting of all possible dichotomies of the labels, as shown at the bottom of table 10.5. In all such codes, the Hamming distance between any pair of rows will be roughly $\overline{K}/2$ (or better), compared to just 2 for the one-against-all code.

In some domains, a binary coding of the labels may already be naturally defined by the nature of the problem. For instance, if classifying phonemes, each class (phoneme) may be naturally described by a set of binary features: voiced or unvoiced, vowel or consonant, fricative or not, and so on. The code Ω then can correspond to the values of each of these binary output features for each phoneme.

Theorem 10.4 formalizes the intuitions above, giving a bound on the training error in terms of the quality of the code as measured by the minimum distance (or symmetric difference) between any pair of codewords. We do not give a proof of the theorem because it will follow as an immediate special case of theorem 10.5 below.

Theorem 10.4 Assuming the notation of algorithm 10.3 and algorithm 10.2 (viewed as a subroutine), let

$$\rho = \min_{\ell_1, \ell_2 \in \mathcal{Y} : \ell_1 \neq \ell_2} |\Omega(\ell_1) \triangle \Omega(\ell_2)|\,. \tag{10.10}$$

Table 10.5
The one-against-all (top) and complete (bottom) coding matrices for a four-class problem

Ω				
a	$+1$	-1	-1	-1
b	-1	$+1$	-1	-1
c	-1	-1	$+1$	-1
d	-1	-1	-1	$+1$

Ω						
a	$+1$	-1	-1	-1	$+1$	$+1$	-1
b	-1	$+1$	-1	-1	$+1$	-1	$+1$
c	-1	-1	$+1$	-1	-1	$+1$	$+1$
d	-1	-1	-1	$+1$	-1	-1	-1

Names for the columns or dichotomies have been omitted. The complete code omits degenerate dichotomies which are all $+1$ or all -1, and also any dichotomy which is the negation of one that has already been included.

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

$$\frac{2\overline{K}}{\rho} \cdot \widehat{\text{hloss}}(H) \leq \frac{2\overline{K}}{\rho} \prod_{t=1}^{T} Z_t$$

for Hamming decoding, and by

$$\frac{\overline{K}}{\rho} \prod_{t=1}^{T} Z_t$$

for loss-based decoding (where $\overline{K} = |\overline{\mathcal{Y}}|$).

We can use theorem 10.4 to improve the bound in equation (10.8) for AdaBoost.MH to that in equation (10.9) when applied to single-label multiclass problems. We apply theorem 10.4 to the code defined by $\Omega(y) = \{y\}$ for all $y \in \mathcal{Y}$. Clearly, $\rho = 2$ in this case. Moreover, we claim that H^1, as defined in equation (10.7), produces predictions identical to those generated by H^{lb} when using loss-based decoding in AdaBoost.MO. This is because

$$\sum_{\overline{y} \in \mathcal{Y}} \exp\left(-\Omega(y, \overline{y})\ F(x, \overline{y})\right) = e^{-F(x,y)} - e^{F(x,y)} + \sum_{\overline{y} \in \mathcal{Y}} e^{F(x,\overline{y})},$$

so that the minimum over y is attained when $F(x, y)$ is maximized. Applying theorem 10.4 now gives the bound in equation (10.9).

Although an improvement, these bounds for AdaBoost.MH are still rather poor in the sense that they depend strongly on the number of classes K, and thus will be weak on

problems with a large number of classes. In fact, when using codes with strong error-correcting properties, theorem 10.4 indicates that there does not need to be an explicit dependence on the number of classes. For instance, if the code Ω is chosen at random (uniformly among all possible codes), then, for large $\overline{K}$, we expect $\rho/\overline{K}$ to approach $\frac{1}{2}$. In this case, the leading coefficients in the bounds of theorem 10.4 approach 4 for Hamming decoding, and 2 for loss-based decoding, independent of the number of classes K in the original label set $\mathcal{Y}$. This suggests that the method may be highly effective on problems with a large number of classes. However, there is an important trade-off here: When a random code Ω is used, the resulting binary problems, which are defined by a random partition of the classes, may be highly unnatural, making it difficult to learn these underlying binary problems.

10.4.2 More General Codes

The output-coding approach described so far requires that every dichotomy of the classes to be learned must involve *all* of the classes. This is potentially a limitation since such binary problems can be exceedingly difficult to learn due to their unnaturalness. For instance, if attempting to optically recognize handwritten digits, it may be very hard to learn to distinguish digits belonging to the set {0, 1, 5, 6, 9} from those belonging to {2, 3, 4, 7, 8}. The problem is that such unnatural, disjunctive concepts are highly complex and difficult to characterize.

For this reason, it is sometimes advantageous to use dichotomies that involve only a *subset* of the classes. For instance, in the example above, we might attempt to learn to distinguish digits in the set {1, 7} from those in the set {0, 6}. A classifier trained for this task would be expected to give accurate predictions only when presented with examples from one of the target classes, in this case, 0, 1, 6, or 7; nothing would be expected of its performance on examples belonging to other classes.

Such a dichotomy involving just a few of the classes is likely to be much simpler, and thus easier to learn. At an extreme, we can consider distinguishing just one class from one other, for instance, distinguishing 3's from 7's, a problem that surely should be easier than the complex dichotomy above involving all 10 classes. When one binary problem is solved for each of the $\binom{K}{2}$ pairs of classes, this leads to the *all-pairs* approach discussed further below.

The output-coding framework outlined above can be extended to accommodate dichotomies involving only a subset of the classes. We saw earlier that the code Ω can be viewed as a matrix of $\{-1, +1\}$ values, an interpretation that we adopt henceforth, abandoning our earlier alternative view of Ω as a mapping to multi-label sets. Furthermore, we now allow entries of Ω to take the value 0 so that as a function, Ω maps $\mathcal{Y} \times \overline{\mathcal{Y}}$ to $\{-1, 0, +1\}$. We interpret the value 0 for entry $\Omega(y, \overline{y})$ to be an indication that class y is irrelevant for dichotomy $\overline{y}$, and thus that a classifier's predictions on examples with this label are immaterial. Such examples are simply ignored during training.

For instance, Ω may be a matrix such as the one at the top of table 10.6. Here, dichotomy 1 asks if an example's class belongs to the set $\{a\}$ or if it belongs to $\{b, d\}$, with examples in

Table 10.6
Another sample coding matrix (top), and its effect on a sample dataset (bottom)

Ω	1	2	3
a	$+1$	-1	-1
b	-1	0	$+1$
c	0	$+1$	-1
d	-1	0	-1

	Dichotomies		
Original	1	2	3
$(x_1, a) \rightarrow$	$(x_1, +1)$	$(x_1, -1)$	$(x_1, -1)$
$(x_2, c) \rightarrow$		$(x_2, +1)$	$(x_2, -1)$
$(x_3, a) \rightarrow$	$(x_3, +1)$	$(x_3, -1)$	$(x_3, -1)$
$(x_4, d) \rightarrow$	$(x_4, -1)$		$(x_4, -1)$
$(x_5, b) \rightarrow$	$(x_5, -1)$		$(x_5, +1)$

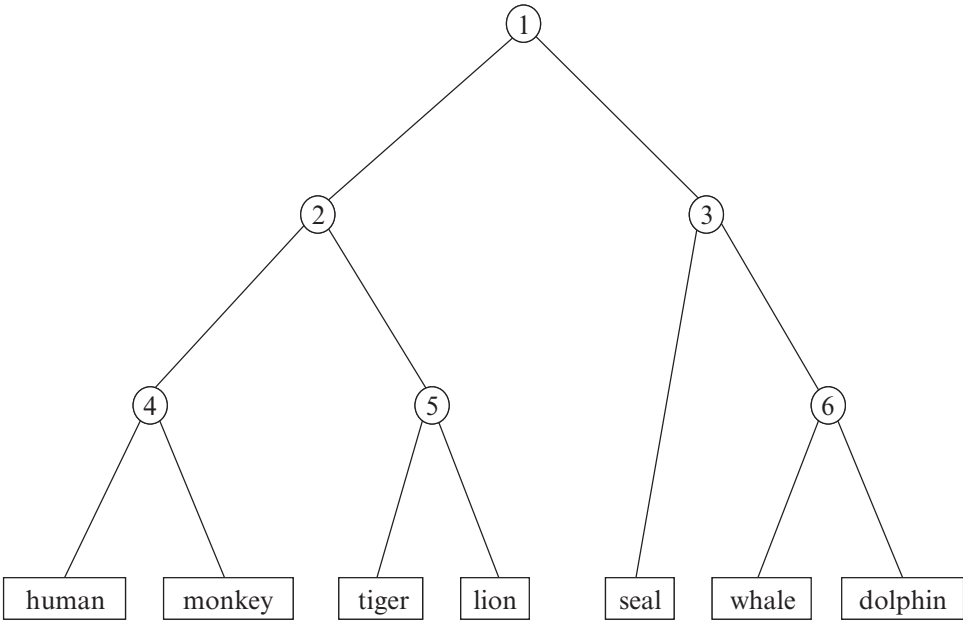
Similar to table 10.4, each multiclass, single-label example in the original dataset is mapped to binary examples in the three dichotomies of this code. Now, however, some of these are omitted from some of the resulting binary datasets.

Table 10.7
The all-pairs coding matrix for a four-class problem

Ω						
a	$+1$	$+1$	$+1$	0	0	0
b	-1	0	0	$+1$	$+1$	0
c	0	-1	0	-1	0	$+1$
d	0	0	-1	0	-1	-1

class c being of no relevance; dichotomy 2 asks if the example's class is in $\{c\}$ or in $\{a\}$, with classes b and d being irrelevant; and dichotomy 3 asks if it is in $\{b\}$ or in $\{a, c, d\}$. The bottom of the figure shows how a multiclass dataset gets mapped to the three binary problems using this code. Note that examples with label c are omitted from the first binary problem, as are examples with label b or d from the second. For instance, example (x_5, b) becomes a negative example for binary problem 1, a positive example for binary problem 3, and is omitted from binary problem 2.

Table 10.7 shows the matrix Ω for the all-pairs code mentioned above for a four-class problem. This code consists of one dichotomy for every pair of classes. Intuitively, such dichotomies should be the easiest and most natural binary problems one could extract from a multiclass problem. On the other hand, when the number of classes is large, the number of dichotomies will be quadratically larger, although the training set for each dichotomy will be relatively small. Also, the error-correcting properties of this code are not so strong.



	1	2	3	4	5	6
human	+1	+1	0	+1	0	0
monkey	+1	+1	0	–1	0	0
tiger	+1	–1	0	0	+1	0
lion	+1	–1	0	0	–1	0
seal	–1	0	+1	0	0	0
whale	–1	0	–1	0	0	+1
dolphin	–1	0	–1	0	0	–1

Figure 10.4
Seven classes arranged naturally in a hierarchy, and a corresponding code based on this hierarchy.

We can sometimes derive codes using known structure among the classes. For instance, it may be that the classes form a natural hierarchy as shown in figure 10.4. In such a case, a code corresponding exactly to this tree structure can be created in which each dichotomy corresponds to an internal node pitting the classes in its left subtree against those in its right subtree, ignoring all others, as shown in the figure.

We can modify AdaBoost.MO to handle such $\{-1, 0, +1\}$-valued codes by ignoring some examples on some of the dichotomies as dictated by the code. In other words, we saw earlier that AdaBoost.MO is a reduction to a binary classification problem in which there is one training instance for each of the $\overline{K}m$ pairs $(x_i, \overline{y})$ for each (x_i, y_i) in the original

training set and each $\overline{y} \in \overline{\mathcal{Y}}$; the binary label assigned to this pair is $\Omega(y_i, \overline{y})$. Now we can follow exactly the same reduction but omit all pairs $(x_i, \overline{y})$ for which $\Omega(y_i, \overline{y}) = 0$. This is equivalent in the context of boosting to setting the initial distribution of such examples to be zero. Thus, mathematically, the only needed modification is in the initialization of D_1. In particular, we let

$$D_1(i, \overline{y}) = \frac{|\Omega(y_i, \overline{y})|}{sm} = \begin{cases} 0 & \text{if } \Omega(y_i, \overline{y}) = 0 \\ 1/(sm) & \text{else,} \end{cases}$$

where s is the *sparsity* measuring the number of non-zeros in the output code when applied to the dataset:

$$s \doteq \frac{1}{m} \sum_{i=1}^{m} \sum_{\overline{y} \in \overline{\mathcal{Y}}} |\Omega(y_i, \overline{y})| = \frac{1}{m} \sum_{i=1}^{m} |S_{y_i}|, \tag{10.11}$$

and where

$$S_y \doteq \{\overline{y} \in \overline{\mathcal{Y}} : \Omega(y, \overline{y}) \neq 0\}.$$

After initializing D_1 in this modified fashion, AdaBoost.MH can be applied just as before, producing a weighted combination of weak hypotheses $F(x, \overline{y})$ whose sign is given by $H(x, \overline{y})$. The decoding methods described above can be generalized to ignore zero entries in the output code. In particular, for Hamming decoding, we redefine H^{ham} to be

$$H^{ham}(x) = \arg\min_{y \in \mathcal{Y}} \sum_{\overline{y} \in S_y} \mathbf{1}\{H(x, \overline{y}) \neq \Omega(y, \overline{y})\}.$$

Likewise, for loss-based decoding, we now have

$$H^{lb}(x) = \arg\min_{y \in \mathcal{Y}} \sum_{\overline{y} \in S_y} \exp(-\Omega(y, \overline{y})\ F(x, \overline{y})).$$

Algorithm 10.4 shows such a generalized version of AdaBoost.MO in which the subroutine call to AdaBoost.MH has been "compiled out." Note that, because we have modified only the distributions D_t, we can continue to use all of the preceding techniques for choosing α_t and for finding weak hypotheses h_t. On the other hand, in many cases it may be possible to implement this same algorithm more efficiently by taking advantage of codes which are very sparse, or which have special structure.

Our analysis of the training error of this method, a direct generalization of theorem 10.4, uses a generalized measure ρ of the minimum Hamming distance between two rows of the code Ω in which entries are ignored if they are 0 in either (or both) of the rows. That is, for distinct rows ℓ_1 and ℓ_2, we first define

Algorithm 10.4
A generalized version of AdaBoost.MO

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$
output code $\Omega : \mathcal{Y} \times \overline{\mathcal{Y}} \to \{-1, 0, +1\}$.
Initialize:
$D_1(i, \overline{y}) = |\Omega(y_i, \overline{y})|/(sm)$ for $i = 1, \ldots, m$ and $\overline{y} \in \overline{\mathcal{Y}}$,
where s is as in equation (10.11).
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \times \overline{\mathcal{Y}} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Aim: select h_t and α_t to minimalize the normalization factor

$$Z_t \doteq \sum_{i=1}^{m} \sum_{\overline{y} \in \overline{\mathcal{Y}}} D_t(i, \overline{y}) \exp(-\alpha_t \, \Omega(y_i, \overline{y}) \, h_t(x_i, \overline{y})).$$

- Update, for $i = 1, \ldots, m$ and $\overline{y} \in \overline{\mathcal{Y}}$:

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

Let

$$F(x, \overline{y}) = \sum_{t=1}^{T} \alpha_t h_t(x, \overline{y})$$

$$H(x, \overline{y}) = \operatorname{sign}(F(x, \overline{y})).$$

Output final hypothesis:

$$H^{ham}(x) = \arg\min_{y \in \mathcal{Y}} \sum_{\overline{y} : \Omega(y, \overline{y}) \neq 0} \mathbf{1}\{H(x, \overline{y}) \neq \Omega(y, \overline{y})\} \quad \text{(Hamming decoding)}$$

or

$$H^{lb}(x) = \arg\min_{y \in \mathcal{Y}} \sum_{\overline{y} : \Omega(y, \overline{y}) \neq 0} \exp(-\Omega(y, \overline{y}) \; F(x, \overline{y})) \quad \text{(loss-based decoding).}$$

$$T_{\ell_1,\ell_2} \doteq \{\overline{y} \in S_{\ell_1} \cap S_{\ell_2} : \Omega(\ell_1, \overline{y}) \neq \Omega(\ell_2, \overline{y})\}$$

to be the set of nonzero entries where ℓ_1 and ℓ_2 differ. Then ρ is the minimum cardinality of any such set. Also, the empirical Hamming error, specialized to this setting and ignoring zero entries, becomes

$$\widehat{\text{hloss}}(H) \doteq \frac{1}{sm} \sum_{i=1}^{m} \sum_{\overline{y} \in S_{y_i}} \mathbf{1}\{H(x_i, \overline{y}) \neq \Omega(y_i, \overline{y})\}.$$

Note that by the same arguments used in theorem 10.2 applied to this modified reduction to binary, this loss is upper bounded by the exponential loss

$$\frac{1}{sm} \sum_{i=1}^{m} \sum_{\overline{y} \in S_{y_i}} \exp(-\Omega(y, \overline{y})\ F(x, \overline{y})) = \prod_{t=1}^{T} Z_t.$$

Theorem 10.5 Assuming the notation of algorithm 10.4, and given the definitions above, let

$$\rho \doteq \min_{\ell_1,\ell_2 \in \mathcal{Y}: \ell_1 \neq \ell_2} |T_{\ell_1,\ell_2}|.$$

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

$$\frac{2s}{\rho} \widehat{\text{hloss}}(H) \leq \frac{2s}{\rho} \prod_{t=1}^{T} Z_t$$

for Hamming decoding, and by

$$\frac{s}{\rho} \prod_{t=1}^{T} Z_t$$

for loss-based decoding.

Proof We give a unified proof for both Hamming and loss-based decoding. In either case, for a fixed example, let $L(x, y, \overline{y})$ be the relevant loss that would be suffered on example x if the correct label were y on dichotomy (or mapped label) $\overline{y} \in S_y$. Thus, for Hamming decoding, where the relevant loss is Hamming (or misclassification) error,

$$L(x, y, \overline{y}) = \mathbf{1}\{H(x, \overline{y}) \neq \Omega(y, \overline{y})\},$$

and for loss-based decoding, based on exponential loss,

$$L(x, y, \overline{y}) = \exp(-\Omega(y, \overline{y})\ F(x, \overline{y})).$$

Suppose the actual correct label for x is y. Then for either coding scheme, the final classifier H^{ham} or H^{lb} makes a mistake only if

$$\sum_{\overline{y} \in S_\ell} L(x, \ell, \overline{y}) \leq \sum_{\overline{y} \in S_y} L(x, y, \overline{y})$$

for some $\ell \neq y$. This implies that

$$\begin{aligned}
\sum_{\overline{y} \in S_y} L(x, y, \overline{y}) &\geq \tfrac{1}{2} \sum_{\overline{y} \in S_y} L(x, y, \overline{y}) + \tfrac{1}{2} \sum_{\overline{y} \in S_\ell} L(x, \ell, \overline{y}) \\
&\geq \tfrac{1}{2} \sum_{\overline{y} \in S_y \cap S_\ell} (L(x, y, \overline{y}) + L(x, \ell, \overline{y})) \\
&\geq \tfrac{1}{2} \sum_{\overline{y} \in T_{y,\ell}} (L(x, y, \overline{y}) + L(x, \ell, \overline{y})), \qquad (10.12)
\end{aligned}$$

where in the second and third inequalities, we have simply dropped some nonnegative terms.

If $\overline{y} \in T_{y,\ell}$, then $\Omega(\ell, \overline{y}) = -\Omega(y, \overline{y})$. Thus, for Hamming decoding, at least one of $L(x, y, \overline{y})$ or $L(x, \ell, \overline{y})$ will be equal to 1 in this case, which implies that equation (10.12) is at least

$$\frac{|T_{y,\ell}|}{2} \geq \frac{\rho}{2}.$$

Therefore, if M^{ham} is the number of training errors made by H^{ham}, then this argument shows that

$$\begin{aligned}
M^{ham} \cdot \frac{\rho}{2} &\leq \sum_{i=1}^{m} \sum_{\overline{y} \in S_{y_i}} L(x_i, y_i, \overline{y}) \\
&= sm\ \widehat{\text{hloss}}(H) \\
&\leq sm \prod_{t=1}^{T} Z_t,
\end{aligned}$$

which is equivalent to the bound for Hamming decoding stated in the theorem.

For loss-based decoding, because $\Omega(\ell, \overline{y}) = -\Omega(y, \overline{y})$ for $\overline{y} \in T_{y,\ell}$, and because we are using exponential loss, equation (10.12) becomes

$$\frac{1}{2} \sum_{\overline{y} \in T_{y,\ell}} \left(L(x, y, \overline{y}) + \frac{1}{L(x, y, \overline{y})} \right),$$

which is at least $|T_{y,\ell}| \geq \rho$ since $z + 1/z \geq 2$ for all $z > 0$. So again, if M^{lb} is the number of training errors made by H^{lb}, then

$$M^{lb}\rho \leq \sum_{i=1}^{m} \sum_{\overline{y} \in S_{y_i}} L(x_i, y_i, \overline{y})$$
$$= sm \prod_{t=1}^{T} Z_t,$$

giving the bound for loss-based decoding. ∎

Theorem 10.4 can be obtained as an immediate corollary simply by applying theorem 10.5 to an output code Ω with no zero entries. The theorem again formalizes the trade-off between codes in which the codewords are far apart, as measured by ρ/s, against the difficulty of learning the various dichotomies, as measured by the Z_t's.

For the all-pairs reduction, $\rho = 1$ and $s = K - 1$, so that with loss-based decoding, the overall training error is at most

$$(K-1)\prod_{t=1}^{T} Z_t.$$

For a code based on a hierarchy as in figure 10.4, $\rho = 1$ and s is at most the depth of the tree.

Experimentally, loss-based decoding seems to nearly always perform at least as well as Hamming decoding, consistent with the theory. However, the code giving best performance on a particular dataset seems to depend very much on the problem at hand. One-against-all is often satisfactory, but not always the best. For instance, on the 26-class "letter" benchmark dataset used elsewhere in this book (such as in section 1.3), the test error rates shown in table 10.8 were obtained using decision stumps as weak hypotheses. On this dataset, all-pairs is far superior to one-against-all, while a random code (with no zero entries) does

Table 10.8
Percent test error rates for various coding and decoding schemes on two benchmark datasets

	Letter		Soybean-Large	
	Hamming	Loss-based	Hamming	Loss-based
One-against-All	27.7	14.6	8.2	7.2
All-Pairs	7.8	7.1	9.0	8.8
Random	30.9	28.3	5.6	4.8

worse than either. On the other hand, on the "soybean-large" benchmark dataset, which has 19 classes, the best results are obtained using a random code, and the worst with the all-pairs reduction.

Summary

In summary, we have seen in this chapter that there are many ways of extending binary AdaBoost when confronting a multiclass learning task. When using a relatively strong base learner, the most straightforward extension, AdaBoost.M1, can be used. For weaker base learners, the multiclass problem must be reduced to multiple binary problems. There are numerous ways of devising such a reduction, and we have discussed several specific and general strategies and analyses. These include AdaBoost.MH, which can be used not only for multiclass but also for multi-label data, as well as AdaBoost.MO, which can be used across a very broad range of reductions or codes.

Bibliographic Notes

The AdaBoost.M1 algorithm and analysis of section 10.1 are due to Freund and Schapire [95], as are the experiments reported at the end of that section and in figure 10.1 [93]. The AdaBoost.MH algorithm and analysis of section 10.2 are due to Schapire and Singer [205]. The experiments in section 10.3 were conducted by Schapire and Singer [206] on a task and with data developed by Gorin and others [109, 110, 189].

The results and methods in section 10.4.1 are from Schapire and Singer [205], and are based directly on the error-correcting output coding technique of Dietterich and Bakiri [70]. The generalization in section 10.4.2 is essentially from Allwein, Schapire, and Singer [6], although here we have included somewhat improved versions of AdaBoost.MO and its analysis in theorem 10.5. Some similar, though more specialized, results were given earlier by Guruswami and Sahai [114]. The all-pairs approach was studied previously by Friedman [99] and Hastie and Tibshirani [119]. The results in table 10.8 are excerpted from far more extensive experiments reported by Allwein, Schapire, and Singer [6], who also gave results on the generalization error of some of the methods studied in this chapter. More general decoding schemes and improved analyses are given, for instance, by Klautau, Jevtić, and Orlitsky [136], and Escalera, Pujol, and Radeva [83].

Other approaches for extending boosting to the multiclass setting have been proposed, for instance, by Schapire [200], Abe, Zadrozny, and Langford [2], Eibl and Pfeiffer [81], Zhu et al. [238], and Mukherjee and Schapire [173]. See also Beygelzimer, Langford, and Ravikumar's [19] more general work on reducing multiclass to binary.

Some of the exercises in this chapter are based on material from [6].

Exercises

10.1 Consider a modification of AdaBoost.M1 (algorithm 10.1) in which the algorithm is *not* forced to halt when $\epsilon_t \geq \frac{1}{2}$, but is simply allowed to proceed. Assume the weak learner is exhaustive, returning on each round the weak classifier $h \in \mathcal{H}$ with minimum weighted error. Suppose on some round t that $\epsilon_t > \frac{1}{2}$.

a. For this modified version of AdaBoost.M1, explain what will happen on all subsequent rounds $t+1, t+2, \ldots$.

b. Under these conditions, how will the resulting combined classifiers differ for the modified and unmodified versions?

10.2 AdaBoost.Mk (algorithm 10.5) is a generalization of AdaBoost.M1 that relaxes the requirement that the weighted errors of the weak classifiers be smaller than $\frac{1}{2}$, but that provides correspondingly weaker guarantees on performance. The algorithm takes an integer parameter $k \geq 1$ (with $k = 1$ corresponding to AdaBoost.M1). The setting of α_t will be discussed shortly. For a real-valued function $f : \mathcal{Y} \to \mathbb{R}$, we use in the algorithm the notation arg k-max$_{y \in \mathcal{Y}} f(y)$ to stand for the top k elements of $\mathcal{Y}$ when ordered by f, that is, a set $A \subseteq \mathcal{Y}$ with $|A| = k$ and for which $f(y) \geq f(y')$ for all $y \in A$ and $y' \notin A$. (If more than one set A satisfies this condition, we allow one to be chosen arbitrarily.) Thus, $H(x)$ returns the top k labels as ordered by the weighted votes of the weak classifiers.

a. Show that

$$\frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\{y_i \notin H(x_i)\} \leq \prod_{t=1}^{T} Z_t.$$

b. Assume that $\epsilon_t < k/(k+1)$ for all t. Show how to choose α_t so that the fraction of training examples for which $y_i \notin H(x_i)$ is at most

$$\exp\left(-\sum_{t=1}^{T} \mathrm{RE}_b\left(\frac{k}{k+1} \;\|\; \epsilon_t\right)\right).$$

c. Conclude that if the weighted accuracy of each weak classifier is at least $1/(k+1) + \gamma$, then after T rounds, at least a fraction $1 - e^{-2\gamma^2 T}$ of the training examples i will have the correct label y_i ranked among the top k, that is, included in the set $H(x_i)$.

10.3 In this exercise and exercise 10.4, we will see one way of generalizing the margins analysis of chapter 5 to the current multiclass setting, specifically, to AdaBoost.MO using loss-based decoding as in algorithm 10.4. For simplicity, assume the weak hypotheses h_t are chosen from a finite space $\mathcal{H}$, and all have range $\{-1, +1\}$. We also assume Ω includes no zero entries, and that (without loss of generality) $\alpha_t \geq 0$ for all t. The convex hull of $\mathcal{H}$,

Algorithm 10.5
AdaBoost.Mk, a generalization of AdaBoost.M1

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$
parameter $k \geq 1$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \mathcal{Y}$.
- Aim: select h_t to minimalize the weighted error:

$$\epsilon_t \doteq \mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i] .$$

- If $\epsilon_t \geq k/(k+1)$, then set $T = t - 1$, and exit loop.
- Choose $\alpha_t > 0$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-k\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

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

Output the final hypothesis:

$$H(x) = \arg k\text{-}\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \alpha_t \mathbf{1}\{h_t(x) = y\} .$$

$\mathrm{co}(\mathcal{H})$, is the same as in equation (5.4), except that the functions involved are defined on the domain $\mathcal{X} \times \overline{\mathcal{Y}}$ rather than $\mathcal{X}$.

For $f \in \mathrm{co}(\mathcal{H})$, $\eta > 0$, and $(x, y) \in \mathcal{X} \times \mathcal{Y}$, let

$$\nu_{f,\eta}(x, y) \doteq -\frac{1}{\eta} \ln \left(\frac{1}{K} \sum_{\overline{y} \in \overline{\mathcal{Y}}} \exp\left(-\eta \Omega(y, \overline{y}) f(x, \overline{y})\right) \right).$$

We define the margin of labeled example (x, y) with respect to f, η to be

$$\mathcal{M}_{f,\eta}(x, y) \doteq \frac{1}{2} \left(\nu_{f,\eta}(x, y) - \max_{\ell \neq y} \nu_{f,\eta}(x, \ell) \right).$$

a. Show that $\mathcal{M}_{f,\eta}(x, y) \in [-1, +1]$. Also, for an appropriate choice of f and η, show that $\mathcal{M}_{f,\eta}(x, y) \leq 0$ if and only if H^{lb} misclassifies (x, y) (where, as usual, we count a tie in the "arg min" used to compute H^{lb} as a misclassification).

Let $f \in \text{co}(\mathcal{H})$ and let $\theta > 0$ be fixed. Let n be a (fixed) positive integer, and let $\mathcal{A}_n, \tilde{f}, \tilde{h}_1, \ldots, \tilde{h}_n$ be as in the proof of theorem 5.1 (but with modified domain $\mathcal{X} \times \overline{\mathcal{Y}}$). We also adopt the notation $\mathbf{Pr}_S[\cdot]$, $\mathbf{Pr}_{\mathcal{D}}[\cdot]$, $\mathbf{Pr}_{\tilde{f}}[\cdot]$, and so on from that proof, where S is the training set and $\mathcal{D}$ is the true distribution over $\mathcal{X} \times \mathcal{Y}$.

b. For fixed $x \in \mathcal{X}$, show that

$$\mathbf{Pr}_{\tilde{f}}\left[\exists \overline{y} \in \overline{\mathcal{Y}} : \left| f(x, y) - \tilde{f}(x, y) \right| \geq \frac{\theta}{4}\right] \leq \beta_n$$

where $\beta_n \doteq 2\overline{K} e^{-n\theta^2/32}$.

In what follows, you can use (without proof) the following technical fact: Let

$$\mathcal{E}_\theta \doteq \left\{ \frac{4 \ln \overline{K}}{i\theta} : i = 1, \ldots, \left\lceil \frac{8 \ln \overline{K}}{\theta^2} \right\rceil \right\}.$$

For any $\eta > 0$, let $\hat{\eta}$ be the closest value in $\mathcal{E}_\theta$ to η. Then for all $f \in \text{co}(\mathcal{H})$ and for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$,

$$\left| \nu_{f,\eta}(x, y) - \nu_{f,\hat{\eta}}(x, y) \right| \leq \frac{\theta}{4}.$$

c. Let $\eta > 0$, and let $\hat{\eta} \in \mathcal{E}_\theta$ be as above. Suppose for some $x \in \mathcal{X}$ that $|f(x, \overline{y}) - \tilde{f}(x, \overline{y})| \leq \theta/4$ for all $\overline{y} \in \overline{\mathcal{Y}}$. Show the following for all $y \in \mathcal{Y}$:

i. $|\nu_{f,\eta}(x, y) - \nu_{\tilde{f},\eta}(x, y)| \leq \theta/4$.

ii. $|\nu_{f,\eta}(x, y) - \nu_{\tilde{f},\hat{\eta}}(x, y)| \leq \theta/2$.

[*Hint:* Prove and then use the fact that $\left(\sum_i a_i\right) / \left(\sum_i b_i\right) \leq \max_i (a_i/b_i)$ for any positive numbers $a_1, \ldots, a_n; b_1, \ldots, b_n$.]

d. For any distribution P over pairs (x, y), show that

$$\mathbf{Pr}_{P,\tilde{f}}\left[\left| \mathcal{M}_{f,\eta}(x, y) - \mathcal{M}_{\tilde{f},\hat{\eta}}(x, y) \right| \geq \frac{\theta}{2}\right] \leq \beta_n.$$

e. Let

$$\varepsilon_n \doteq \sqrt{\frac{\ln\left[|\mathcal{E}_\theta| \cdot |\mathcal{H}|^n/\delta\right]}{2m}}.$$

Show that with probability at least $1 - \delta$ over the choice of the random training set, for all $\tilde{f} \in \mathcal{A}_n$, and for all $\hat{\eta} \in \mathcal{E}_\theta$,

$$\mathbf{Pr}_{\mathcal{D}}\left[\mathcal{M}_{\tilde{f},\hat{\eta}}(x, y) \leq \frac{\theta}{2}\right] \leq \mathbf{Pr}_{S}\left[\mathcal{M}_{\tilde{f},\hat{\eta}}(x, y) \leq \frac{\theta}{2}\right] + \varepsilon_n.$$

(Note that θ and n are *fixed*.)

f. Show that with probability at least $1 - \delta$, for all $f \in \text{co}(\mathcal{H})$, and for all $\eta > 0$,

$$\mathbf{Pr}_{\mathcal{D}}\big[\mathcal{M}_{f,\eta}(x, y) \leq 0\big] \leq \mathbf{Pr}_{S}\big[\mathcal{M}_{f,\eta}(x, y) \leq \theta\big] + 2\beta_n + \varepsilon_n.$$

For an appropriate choice of n, we can now obtain a result analogous to theorem 5.1 (you do not need to show this).

10.4 Continuing exercise 10.3, let

$$\epsilon_t \doteq \mathbf{Pr}_{(i,\overline{y})\sim D_t}[h_t(x_i, \overline{y}) \neq \Omega(y_i, \overline{y})]$$

be the weighted error of h_t, which we assume without loss of generality is at most $\frac{1}{2}$, and let α_t be chosen as in equation (10.4). Let f and η be chosen as in exercise 10.3(a), and let $\theta > 0$.

a. Suppose, for some $(x, y) \in \mathcal{X} \times \mathcal{Y}$, that $\mathcal{M}_{f,\eta}(x, y) \leq \theta$. For $\overline{y} \in \overline{\mathcal{Y}}$ and $\ell \in \mathcal{Y}$, let

$$z(\overline{y}) \doteq \eta\Omega(y, \overline{y}) f(x, \overline{y}) - \eta\theta$$

$$z_\ell(\overline{y}) \doteq \eta\Omega(\ell, \overline{y}) f(x, \overline{y}) + \eta\theta.$$

Show that

i. $\displaystyle\sum_{\overline{y}\in\overline{\mathcal{Y}}} e^{-z(\overline{y})} \geq \sum_{\overline{y}\in\overline{\mathcal{Y}}} e^{-z_\ell(\overline{y})}$ for some $\ell \neq y$.

ii. $\displaystyle\sum_{\overline{y}\in\overline{\mathcal{Y}}} e^{-z(\overline{y})} \geq \rho$ where ρ is as in equation (10.10).

b. Let $\gamma_t \doteq \frac{1}{2} - \epsilon_t$. Prove that the fraction of training examples i for which $\mathcal{M}_{f,\eta}(x_i, y_i) \leq \theta$ is at most

$$\frac{\overline{K}}{\rho} \cdot \prod_{t=1}^{T} \sqrt{(1 + 2\gamma_t)^{1+\theta}(1 - 2\gamma_t)^{1-\theta}}.$$

10.5 When using AdaBoost.MO with the all-pairs coding matrix, each dichotomy $\overline{y}$ is identified with an unordered pair of distinct labels, that is,

$$\overline{\mathcal{Y}} = \{\{\ell_1, \ell_2\} : \ell_1, \ell_2 \in \mathcal{Y}, \ell_1 \neq \ell_2\}.$$

Suppose each weak hypothesis $h_t : \mathcal{X} \times \overline{\mathcal{Y}} \to \mathbb{R}$ can be written in the form

$$h_t(x, \{\ell_1, \ell_2\}) = \frac{\Omega(\ell_1, \{\ell_1, \ell_2\})}{2} \cdot \left(\tilde{h}_t(x, \ell_1) - \tilde{h}_t(x, \ell_2)\right) \tag{10.13}$$

for some $\tilde{h}_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.

a. In equation (10.13), ℓ_1 and ℓ_2 are treated symmetrically on the left, but appear not to be so treated on the right. Show that the right-hand side of equation (10.13) is in fact equal to the same expression if ℓ_1 and ℓ_2 are swapped.

b. Show that if loss-based decoding is used, then

$$H^{lb}(x) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \alpha_t \tilde{h}_t(x, y).$$

10.6 Suppose the label set $\mathcal{Y} = \{0, \ldots, K-1\}$, and that we apply AdaBoost.MO with $\overline{\mathcal{Y}} \doteq \{1, \ldots, K-1\}$ and

$$\Omega(y, \overline{y}) \doteq \begin{cases} +1 & \text{if } \overline{y} \leq y \\ -1 & \text{else.} \end{cases}$$

Using the notation from algorithm 10.3, suppose it happens that the computed function F is monotone in the sense that $F(x, \overline{y}_1) \geq F(x, \overline{y}_2)$ if $\overline{y}_1 \leq \overline{y}_2$.

a. Show that the two decoding methods are equivalent in this case, that is, $H^{ham} \equiv H^{lb}$ (assuming ties in their respective arg mins are broken in the same way).

b. Show that

$$\frac{1}{m} \sum_{i=1}^{m} \frac{H^{lb}(x_i) - y_i}{K-1} \leq \prod_{t=1}^{T} Z_t.$$

c. Suppose each h_t has the form

$$h_t(x, \overline{y}) = \begin{cases} +1 & \text{if } \overline{y} \leq \tilde{h}_t(x) \\ -1 & \text{else} \end{cases}$$

for some $\tilde{h}_t : \mathcal{X} \to \mathcal{Y}$, and assume also that $\alpha_t \geq 0$ for all t. Show that $H^{lb}(x)$ is a *weighted median* of the $\tilde{h}_t(x)$ values with weights α_t. (A weighted median of real numbers $v_1, \ldots, v_n$ with nonnegative weights $w_1, \ldots, w_n$ is any number v for which $\sum_{i:v_i<v} w_i \leq \frac{1}{2}\sum_{i=1}^{n} w_i$ and $\sum_{i:v_i>v} w_i \leq \frac{1}{2}\sum_{i=1}^{n} w_i$.)

10.7 AdaBoost.MO is designed to find F to minimize $\frac{1}{m}\sum_{i=1}^{m} L(F, (x_i, y_i))$ where

$$L(F, (x, y)) \doteq \sum_{\overline{y} \in S_y} \exp\left(-\Omega(y, \overline{y}) F(x, \overline{y})\right).$$

a. Let Ω be any code (possibly with zero entries), and let p be any distribution on $\mathcal{X} \times \mathcal{Y}$. For simplicity, assume $p(y|x) > 0$ for all x, y. Among *all* functions $F : \mathcal{X} \times \overline{\mathcal{Y}} \to \mathbb{R}$, find one that minimizes the expected loss $\mathbf{E}_{(x,y)\sim p}[L(F, (x, y))]$.

b. Let $\mathcal{Y}$, $\overline{\mathcal{Y}}$, Ω, and F be as in exercise 10.6. For any x, find a conditional probability distribution $p(y|x)$ such that $F(x, \cdot)$ minimizes the conditional expected loss

$$\mathbf{E}_{y\sim p(\cdot|x)}[L(F, (x, y))] .$$

Your answer should be in closed form, and expressed in terms of F.