Journal of Machine Learning Research 1 (2000) 113-141 Submitted 5/00; Published 12/00

Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers

Erin L. Allwein EALLWEIN@SWRI.ORG
Southwest Research Institute
6220 Culebra Road
San Antonio, TX 78228

Robert E. Schapire SCHAPIRE@RESEARCH.ATT.COM
AT&T Labs − Research
Shannon Laboratory
180 Park Avenue, Room A203
Florham Park, NJ 07932

Yoram Singer SINGER@CS.HUJI.AC.IL
School of Computer Science & Engineering
Hebrew University, Jerusalem 91904, Israel

Editor: Leslie Pack Kaelbling

Abstract

We present a unifying framework for studying the solution of multiclass categorization problems by reducing them to multiple binary problems that are then solved using a margin-based binary learning algorithm. The proposed framework unifies some of the most popular approaches in which each class is compared against all others, or in which all pairs of classes are compared to each other, or in which output codes with error-correcting properties are used. We propose a general method for combining the classifiers generated on the binary problems, and we prove a general empirical *multiclass* loss bound given the empirical loss of the individual *binary* learning algorithms. The scheme and the corresponding bounds apply to many popular classification learning algorithms including support-vector machines, AdaBoost, regression, logistic regression and decision-tree algorithms. We also give a multiclass generalization error analysis for general output codes with AdaBoost as the binary learner. Experimental results with SVM and AdaBoost show that our scheme provides a viable alternative to the most commonly used multiclass algorithms.

1. Introduction

Many supervised machine learning tasks can be cast as the problem of assigning elements to a finite set of classes or categories. For example, the goal of optical character recognition (OCR) systems is to determine the digit value $(0, \dots, 9)$ from its image. The number of applications that require multiclass categorization is immense. A few examples for such applications are text and speech categorization, natural language processing tasks such as part-of-speech tagging, and gesture and object recognition in machine vision.

In designing machine learning algorithms, it is often easier first to devise algorithms for distinguishing between only two classes. Some machine learning algorithms, such as C4.5 (Quinlan, 1993) and CART (Breiman, Friedman, Olshen, & Stone, 1984), can then be naturally extended to handle the multiclass case. For other algorithms, such as AdaBoost (Freund & Schapire, 1997;

Schapire & Singer, 1999) and the support-vector machines (SVM) algorithm (Vapnik, 1995; Cortes & Vapnik, 1995), a direct extension to the multiclass case may be problematic. Typically, in such cases, the multiclass problem is reduced to multiple binary classification problems that can be solved separately. Connectionist models (Rumelhart, Hinton, & Williams, 1986), in which each class is represented by an output neuron, are a notable example: each output neuron serves as a discriminator between the class it represents and all of the other classes. Thus, this training algorithm is based on a reduction of the multiclass problem to k binary problems, where k is the number of classes.

There are many ways to reduce a multiclass problem to multiple binary classification problems. In the simple approach mentioned above, each class is compared to all others. Hastie and Tibshirani (1998) suggest a different approach in which all pairs of classes are compared to each other. Dietterich and Bakiri (1995) presented a general framework in which the classes are partitioned into opposing subsets using error-correcting codes. For all of these methods, after the binary classification problems have been solved, the resulting set of binary classifiers must then be combined in some way. In this paper, we study a general framework, which is a simple extension of Dietterich and Bakiri's framework, that unifies all of these methods of reducing a multiclass problem to a binary problem.

We pay particular attention to the case in which the binary learning algorithm is one that is based on the *margin* of a training example. Roughly speaking, the margin of a training example is a number that is positive if and only if the example is correctly classified by a given classifier and whose magnitude is a measure of confidence in the prediction. Several well known algorithms work directly with margins. For instance, the SVM algorithm (Vapnik, 1995; Cortes & Vapnik, 1995) attempts to maximize the minimum margin of any training example. There are many more algorithms that attempt to minimize some loss function of the margin. AdaBoost (Freund & Schapire, 1997; Schapire & Singer, 1999) is one example: it can be shown that AdaBoost is a greedy procedure for minimizing an exponential loss function of the margins. In Section 2, we catalog many other algorithms that also can be viewed as margin-based learning algorithms, including regression, logistic regression and decision-tree algorithms.

The simplest method of combining the binary classifiers (which we call *Hamming decoding*) ignores the loss function that was used during training as well as the confidences attached to predictions made by the classifier. In Section 3, we give a new and general technique for combining classifiers that does not suffer from either of these defects. We call this method *loss-based decoding*.

We next prove some of the theoretical properties of these methods in Section 4. In particular, for both of the decoding methods, we prove general bounds on the training error on the multiclass problem in terms of the empirical performance on the individual binary problems. These bounds indicate that loss-based decoding is superior to Hamming decoding. Also, these bounds depend on the manner in which the multiclass problem has been reduced to binary problems. For the one-against-all approach, our bounds are linear in the number of classes, but for a reduction based on random partitions of the classes, the bounds are *independent* of the number of classes. These results generalize more specialized bounds proved by Schapire and Singer (1999) and by Guruswami and Sahai (1999).

In Section 5, we prove a bound on the generalization error of our method when the binary learner is AdaBoost. In particular, we generalize the analysis of Schapire et al. (1998), expressing a bound on the generalization error in terms of the training-set margins of the combined multiclass classifier, and showing that boosting, when used in this way, tends to aggressively increase the margins of the training examples.

Finally, in Section 6, we present experiments using SVM and AdaBoost with a variety of multiclass-to-binary reductions. These results show that, as predicted by our theory, loss-based decoding is almost always better than Hamming decoding. Further, the results show that the most commonly used one-against-all reduction is easy to beat, but that the best method seems to be problem-dependent.

2. Margin-based Learning Algorithms

We study methods for handling multiclass problems using a general class of binary algorithms that attempt to minimize a margin-based loss function. In this section, we describe that class of learning algorithms with several examples.

A *binary margin-based learning algorithm* takes as input binary labeled training examples $(x_1, y_1), \ldots, (x_m, y_m)$ where the *instances* x_i belong to some domain $\mathcal{X}$ and the *labels* $y_i \in \{-1, +1\}$. Such a learning algorithm uses the data to generate a real-valued function or *hypothesis* $f : \mathcal{X} \to \mathbb{R}$ where f belongs to some *hypothesis space* $\mathcal{F}$. The *margin* of an example (x, y) with respect to f is $yf(x)$. Note that the margin is positive if and only if the sign of $f(x)$ agrees with y. Thus, if we interpret the sign of $f(x)$ as its prediction on x, then

$$\frac{1}{m}\sum_{i=1}^{m} [\![y_i f(x_i) \leq 0]\!]$$

is exactly the training error of f, where, in this case, we count a zero output ($f(x_i) = 0$) as a mistake. (Here and throughout this paper, $[\![\pi]\!]$ is 1 if predicate π holds and 0 otherwise.)

Although minimization of the training error may be a worthwhile goal, in its most general form the problem is intractable (see for instance the work of Höffgen and Simon (1992)). It is therefore often advantageous to instead minimize some other nonnegative *loss function* of the margin, that is, to minimize

$$\frac{1}{m}\sum_{i=1}^{m} L(y_i f(x_i)) \tag{1}$$

for some loss function $L : \mathbb{R} \to [0, \infty)$. Different choices of the loss function L and different algorithms for (approximately) minimizing Eq. (1) over some hypothesis space lead to various well-studied learning algorithms. Below we list several examples. In the present work, we are not particularly concerned with the method used to achieve a small empirical loss since we will use these algorithms later in the paper as "black boxes." We focus instead on the loss function itself whose properties will allow us to prove our main theorem on the effectiveness of output coding methods for multiclass problems.

Support-vector Machines. For training data that may not be linearly separable, the support-vector machines (SVM) algorithm (Vapnik, 1995; Cortes & Vapnik, 1995) seeks a linear classifier $f : \mathbb{R}^n \to \mathbb{R}$ of the form $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ that minimizes the objective function

$$\frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m} \xi_i \ ,$$

for some parameter C, subject to the linear constraints

$$y_i((x_i \cdot \mathbf{w}) + b) \geq 1 - \xi_i, \ \ \xi_i \geq 0 \ .$$

Put another way, the SVM solution for $\mathbf{w}$ is the minimizer of the regularized empirical loss function

$$\frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m}(1 - y_i((\mathbf{w}\cdot x_i) + b))_+ \ ,$$

where $(z)_+ = \max\{z, 0\}$. (For a more formal treatment see, for instance, the work of Schölkopf et al. (1998).) Although the role of the L_2 norm of $\mathbf{w}$ in the objective function is fundamental in order for SVM to work, the analysis presented in the next section (and the corresponding multiclass algorithm) depends only on the loss function (which is a function of the margins). Thus, SVM can be viewed here as a binary margin-based learning algorithm which seeks to achieve small empirical risk for the loss function $L(z) = (1 - z)_+$.

AdaBoost. The algorithm AdaBoost (Freund & Schapire, 1997; Schapire & Singer, 1999) builds a hypothesis f that is a linear combination of *weak* or *base hypotheses* h_t:

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

The hypothesis f is built up in a series of rounds on each of which an h_t is selected by a *weak* or *base learning algorithm* and $\alpha_t \in \mathbb{R}$ is then chosen. It has been observed by Breiman (1997a, 1997b) and other authors (Collins, Schapire, & Singer, 2000; Friedman, Hastie, & Tibshirani, 2000; Mason, Baxter, Bartlett, & Frean, 1999; Rätsch, Onoda, & Müller, to appear; Schapire & Singer, 1999) that the h_t's and α_t's are effectively being greedily chosen so as to minimize

$$\frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)}.$$

Thus, AdaBoost is a binary margin-based learning algorithm in which the loss function is $L(z) = e^{-z}$.

AdaBoost with randomized predictions. In a little studied variant of AdaBoost (Freund & Schapire, 1997), we allow AdaBoost to output randomized predictions in which the predicted label of a new example x is chosen randomly to be $+1$ with probability $1/(1 + e^{-2f(x)})$. The loss suffered then is the probability that the randomly chosen predicted label disagrees with the correct label y. Let $p(x) \stackrel{\text{def}}{=} 1/(1 + e^{-2f(x)})$. Then the loss is $p(x)$ if $y = -1$ and $1 - p(x)$ if $y = +1$. Using a simple algebraic manipulation, the loss can be shown to be $1/(1+e^{2yf(x)})$. So for this variant of AdaBoost, we set $L(z) = 1/(1 + e^{2z})$. However, in this case, note that the learning algorithm is not directly attempting to minimize this loss (it is instead minimizing the exponential loss described above).

Regression. There are various algorithms, such as neural networks and least squares regression, that attempt to minimize the squared error loss function $(y - f(x))^2$. When the y's are in $\{-1, +1\}$, this function can be rewritten as

$$\begin{aligned}(y - f(x))^2 &= y^2(y - f(x))^2 \\ &= (yy - yf(x))^2 \\ &= (1 - yf(x))^2.\end{aligned}$$

Thus, for binary problems, minimizing squared error fits our framework where $L(z) = (1 - z)^2$.

Logistic regression. In logistic regression and related methods such as Iterative Scaling (Csiszár & Tusnády, 1984; Della Pietra, Della Pietra, & Lafferty, 1997; Lafferty, 1999), and LogitBoost (Friedman et al., 2000), one posits a logistic model for estimating the conditional probability of a positive label:

$$\Pr[y = +1|x] = \frac{1}{1 + e^{-2f(x)}}.$$

One then attempts to maximize the likelihood of the labels in the sample, or equivalently, to minimize the log loss

$$-\log(\Pr[y|x]) = \log(1 + e^{-2yf(x)}).$$

Thus, for logistic regression and related methods, we take $L(z) = \log(1 + e^{-2z})$.

Decision trees. The most popular decision tree algorithms can also be naturally linked to loss functions. For instance, Quinlan's C4.5 (1993), in its simplest form, for binary classification problems, splits decision nodes in a manner to greedily minimize

$$\sum_{\text{leaf}\, j} \left(p_j^+ \ln\left(\frac{p_j^- + p_j^+}{p_j^+}\right) + p_j^- \ln\left(\frac{p_j^- + p_j^+}{p_j^-}\right)\right) \tag{2}$$

where p_j^+ and p_j^- are the fraction of positive and negative examples reaching leaf j, respectively. The prediction at leaf j is then $\mathrm{sign}(p_j^+ - p_j^-)$. Viewed differently, imagine a decision tree that instead outputs a real number f_j at each leaf with the intention of performing logistic regression as above. Then the empirical loss associated with logistic regression is

$$\sum_{\text{leaf}\, j} \left(p_j^+ \ln(1 + e^{-2f_j}) + p_j^- \ln(1 + e^{2f_j})\right).$$

This is minimized, over choices of f_j, when $f_j = (1/2)\ln(p_j^+/p_j^-)$. Plugging in this choice gives exactly Eq. (2), and thresholding f_j gives the hard prediction rule used earlier. Thus, C4.5, in this simple form, can be viewed as a margin-based learning algorithm that is naturally linked to the loss function used in logistic regression.

By similar reasoning, CART (Breiman et al., 1984), which splits using the Gini index, can be linked to the square loss function, while Kearns and Mansour's (1996) splitting rule can be linked to the exponential loss used by AdaBoost.

The analysis we present in the next section might also hold for other algorithms that tacitly employ a function of the margin. For instance, Freund's BrownBoost algorithm (1999) implicitly uses an instance potential function that satisfies the condition we impose on L. Therefore, it can also be combined with output coding and used to solve multiclass problems. To conclude this section, we plot in Figure 1 some of the loss functions discussed above.

3. Output Coding for Multiclass Problems

In the last section, we discussed margin-based algorithms for learning binary problems. Suppose now that we are faced with a multiclass learning problem in which each label y is chosen from a set $\mathcal{Y}$ of cardinality $k > 2$. How can a binary margin-based learning algorithm be modified to handle a k-class problem?

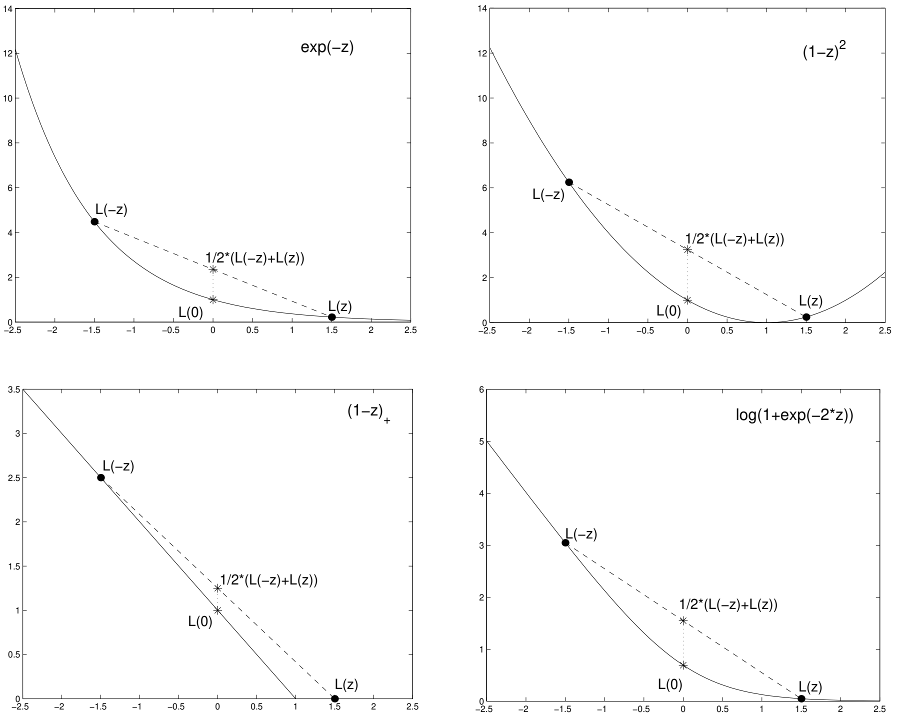

Figure 1: Some of the margin-based loss functions discussed in the paper: the exponential loss used by AdaBoost (top left); the square loss used in least-squares regression (top right); the “hinge” loss used by support-vector machines (bottom left); and the logistic loss used in logistic regression (bottom right).

Several solutions have been proposed for this question. Many involve reducing the multiclass problem, in one way or another, to a set of binary problems. For instance, perhaps the simplest approach is to create one binary problem for each of the k classes. That is, for each $r \in \mathcal{Y}$, we apply the given margin-based learning algorithm to a binary problem in which all examples labeled $y = r$ are considered positive examples and all other examples are considered negative examples. We then end up with k hypotheses that somehow must be combined. We call this the *one-against-all* approach.

Another approach, suggested by Hastie and Tibshirani (1998), is to use the given binary learning algorithm to distinguish each pair of classes. Thus, for each distinct pair $r_1, r_2 \in \mathcal{Y}$, we run the learning algorithm on a binary problem in which examples labeled $y = r_1$ are considered positive, and those labeled $y = r_2$ are negative. All other examples are simply ignored. Again, the $\binom{k}{2}$ hypotheses that are generated by this process must then be combined. We call this the *all-pairs* approach.

A more general suggestion on handling multiclass problems was given by Dietterich and Bakiri (1995). Their idea is to associate each class $r \in \mathcal{Y}$ with a row of a “coding matrix” $\mathbf{M} \in \{-1, +1\}^{k \times \ell}$ for some ℓ. The binary learning algorithm is then run once for each column of the matrix on the induced binary problem in which the label of each example labeled y is mapped to $M(y, s)$. This yields ℓ

hypotheses f_s. Given an example x, we then predict the label y for which row y of matrix $\mathbf{M}$ is "closest" to $(f_1(x), \ldots, f_\ell(x))$. This is the method of *error correcting output codes* (ECOC).

In this section, we propose a unifying generalization of all three of these methods applicable to any margin-based learning algorithm. This generalization is closest to the ECOC approach of Dietterich and Bakiri (1995) but differs in that the coding matrix is taken from the larger set $\{-1, 0, +1\}^{k\times\ell}$. That is, some of the entries $M(r, s)$ may be zero, indicating that we don't care how hypothesis f_s categorizes examples with label r.

Thus, our scheme for learning multiclass problems using a binary margin-based learning algorithm $\mathcal{A}$ works as follows. We begin with a given *coding matrix*

$$\mathbf{M} \in \{-1, 0, +1\}^{k\times\ell} .$$

For $s = 1, \ldots, \ell$, the learning algorithm $\mathcal{A}$ is provided with labeled data of the form $(x_i, M(y_i, s))$ for all examples i in the training set but omitting all examples for which $M(y_i, s) = 0$. The algorithm $\mathcal{A}$ uses this data to generate a hypothesis $f_s : \mathcal{X} \to \mathbb{R}$.

For example, for the one-against-all approach, $\mathbf{M}$ is a $k\times k$ matrix in which all diagonal elements are $+1$ and all other elements are -1. For the all-pairs approach, $\mathbf{M}$ is a $k \times \binom{k}{2}$ matrix in which each column corresponds to a distinct pair (r_1, r_2). For this column, $\mathbf{M}$ has $+1$ in row r_1, -1 in row r_2 and zeros in all other rows.

As an alternative to calling $\mathcal{A}$ repeatedly, in some cases, we may instead wish to add the column index s as a distinguished attribute of the instances received by $\mathcal{A}$, and then learn a *single* hypothesis on this larger learning problem rather than ℓ hypotheses on smaller problems. That is, we provide $\mathcal{A}$ with instances of the form $((x_i, s), M(y_i, s))$ for all training examples i and all columns s for which $M(y_i, s) \neq 0$. Algorithm $\mathcal{A}$ then produces a *single* hypothesis $f : \mathcal{X} \times \{1, \ldots, \ell\} \to \mathbb{R}$. However, for consistency with the preceding approach, we define $f_s(x)$ to be $f(x, s)$. We call these two approaches in which $\mathcal{A}$ is called repeatedly or only once the *multi-call* and *single-call* variants, respectively.

We note in passing that there are no fundamental differences between the single and multi-call variants. Most previous work on output coding employed the multi-call variant due to its simplicity. The single-call variant becomes handy when an implementation of a classification learning algorithm that outputs a single hypothesis of the form $f : \mathcal{X} \times \{1, \ldots, \ell\} \to \mathbb{R}$ is available. We describe experiments with both variants in Section 6.

For either variant, the algorithm $\mathcal{A}$ attempts to minimize the loss L on the induced binary problem(s). Recall that L is a function of the margin of an example so the loss of f_s on an example x_i with induced label $M(y_i, s) \in \{-1, +1\}$ is $L(M(y_i, s)\ f_s(x_i))$. When $M(y_i, s) = 0$, we want to entirely ignore the hypothesis f_s in computing the loss. We can define the loss to be any constant in this case, so, for convenience, we choose the loss to be $L(0)$ so that the loss associated with f_s on example i is $L(M(y_i, s)\ f_s(x_i))$ in all cases.

Thus, the average loss over all choices of s and all examples i is

$$\frac{1}{m\ell} \sum_{i=1}^{m} \sum_{s=1}^{\ell} L(M(y_i, s)\ f_s(x_i)). \tag{3}$$

We call this the *average binary loss* of the hypotheses f_s on the given training set with respect to coding matrix $\mathbf{M}$ and loss L. It is the quantity that the calls to $\mathcal{A}$ have the implicit intention of minimizing. We will see in the next section how this quantity relates to the misclassification error of the final classifier that we build on the original multiclass training set.

Let $\mathbf{M}(r)$ denote row r of M and let $\mathbf{f}(x)$ be the vector of predictions on an instance x:

$$\mathbf{f}(x) = (f_1(x), \ldots, f_\ell(x)).$$

Given the predictions of the f_s's on a test point x, which of the k labels in $\mathcal{Y}$ should be predicted? While several methods of combining the f_s's can be devised, in this paper, we focus on two that are very simple to implement and for which we can analyze the empirical risk of the original multiclass problem. The basic idea of both methods is to predict with the label r whose row $\mathbf{M}(r)$ is "closest" to the predictions $\mathbf{f}(x)$. In other words, predict the label r that minimizes $d(\mathbf{M}(r), \mathbf{f}(x))$ for some distance d. This formulation begs the question, however, of how we measure distance between the two vectors.

One way of doing this is to count up the number of positions s in which the sign of the prediction $f_s(x)$ differs from the matrix entry $M(r,s)$. Formally, this means our distance measure is

$$d_H(\mathbf{M}(r), \mathbf{f}(x)) = \sum_{s=1}^{\ell} \left(\frac{1 - \text{sign}(M(r,s)f_s(x))}{2} \right) \tag{4}$$

where $\text{sign}(z)$ is $+1$ if $z > 0$, -1 if $z < 0$, and 0 if $z = 0$. This is essentially like computing Hamming distance between row $\mathbf{M}(r)$ and the signs of the $f_s(x)$'s. However, note that if either $M(r,s)$ or $f_s(x)$ is zero then that component contributes $1/2$ to the sum. For an instance x and a matrix M, the predicted label $\hat{y} \in \{1, \ldots, k\}$ is therefore

$$\hat{y} = \arg\min_r d_H(\mathbf{M}(r), \mathbf{f}(x)) \ .$$

We call this method of combining the f_s's *Hamming decoding*.

A disadvantage of this method is that it ignores entirely the magnitude of the predictions which can often be an indication of a level of "confidence." Our second method for combining predictions takes this potentially useful information into account, as well as the relevant loss function L which is ignored with Hamming decoding. The idea is to choose the label r that is most consistent with the predictions $f_s(x)$ in the sense that, if example x were labeled r, the total loss on example (x, r) would be minimized over choices of $r \in \mathcal{Y}$. Formally, this means that our distance measure is the total loss on a proposed example (x, r):

$$d_L(\mathbf{M}(r), \mathbf{f}(x)) = \sum_{s=1}^{\ell} L(M(r,s)f_s(x)) \ . \tag{5}$$

Analogous to Hamming decoding, the predicted label $\hat{y} \in \{1, \ldots, k\}$ is

$$\hat{y} = \arg\min_r d_L(\mathbf{M}(r), \mathbf{f}(x)) \ .$$

We call this approach *loss-based decoding*. An illustration of the two decoding methods is given in Figure 2. The figure shows the decoding process for a problem with 4 classes using an output code of length $\ell = 7$. For clarity we denote in the figure the entries of the output code matrix by $+$, $-$ and 0 (instead of $+1$, -1 and 0). Note that in the example, the predicted class of the loss-based decoding (which, in this case, uses exponential loss) is different than that of the Hamming decoding.

We note in passing that the loss-based decoding method for log-loss is the well known and widely used maximum-likelihood decoding which was studied briefly in the context of ECOC by Guruswami and Sahai (1999).

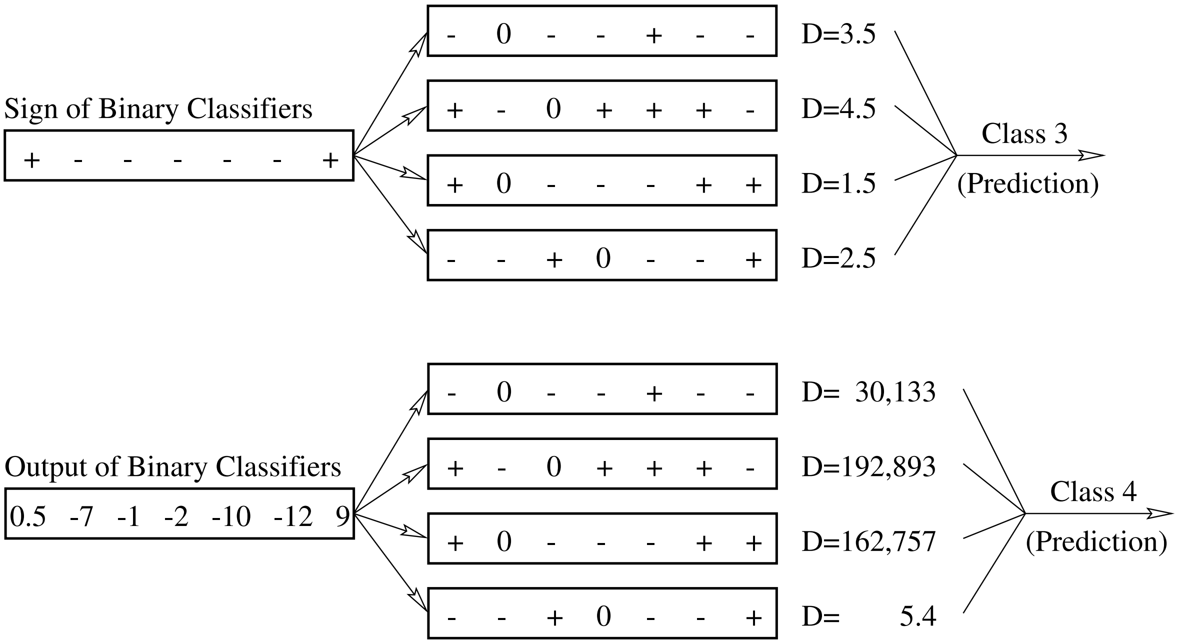

Figure 2: An illustration of the multiclass prediction procedure for Hamming decoding (top) and loss-based decoding (bottom) for a 4-class problem using a code of length 7. The exponential function was used for the loss-based decoding.

4. Analysis of the Training Error

In this section, we analyze the training error of the output coding methods described in the last section. Specifically, we upper bound the training error of the two decoding methods in terms of the average binary loss as defined in Eq. (3), as well as a measure of the minimum distance between any pair of rows of the coding matrix. Here, we use a simple generalization of the Hamming distance for vectors over the set $\{-1, 0, +1\}$. Specifically, we define the distance between two rows $\mathbf{u}, \mathbf{v} \in \{-1, 0, +1\}^\ell$ to be

$$\begin{aligned}
\Delta(\mathbf{u}, \mathbf{v}) &= \sum_{s=1}^{\ell} \begin{cases} 0 & \text{if } u_s = v_s \wedge u_s \neq 0 \wedge v_s \neq 0 \\ 1 & \text{if } u_s \neq v_s \wedge u_s \neq 0 \wedge v_s \neq 0 \\ 1/2 & \text{if } u_s = 0 \vee v_s = 0 \end{cases} \\
&= \sum_{s=1}^{\ell} \frac{1 - u_s v_s}{2} \\
&= \frac{\ell - \mathbf{u} \cdot \mathbf{v}}{2}.
\end{aligned}$$

Our analysis then depends on the minimum distance ρ between pairs of distinct rows:

$$\rho = \min\{\Delta(\mathbf{M}(r_1), \mathbf{M}(r_2)) : r_1 \neq r_2\}. \tag{6}$$

For example, for the one-against-all code, $\rho = 2$. For the all-pairs code, $\rho = (\binom{k}{2} - 1)/2 + 1$, since every two rows r_1, r_2 have exactly one component with opposite signs ($\mathbf{M}(r_1, s) = -\mathbf{M}(r_2, s)$

and $\mathbf{M}(r_1, s) \neq 0$) and for the rest at least one component of the two is 0 ($\mathbf{M}(r_1, s) = 0$ or $\mathbf{M}(r_2, s) = 0$). For a random matrix with components chosen uniformly over either $\{-1, +1\}$ or $\{-1, 0, +1\}$, the *expected* value of $\Delta(\mathbf{M}(r_1), \mathbf{M}(r_2))$ for any distinct pair of rows is exactly $\ell/2$.

Intuitively, the larger ρ, the more likely it is that decoding will "correct" for errors made by individual hypotheses. This was Dietterich and Bakiri's (1995) insight in suggesting the use of output codes with error-correcting properties. This intuition is reflected in our analysis in which a larger value of ρ gives a better upper bound on the training error. In particular, Theorem 1 states that the training error is at most ℓ/ρ times worse than the average binary loss of the combined hypotheses (after scaling the loss by $L(0)$). For the one-against-all matrix, $\ell/\rho = \ell/2 = k/2$ which can be large if the number of classes is large. On the other hand, for the all-pairs matrix or for a random matrix, ℓ/ρ is close to the constant 2, independent of k.

We begin with an analysis of loss-based decoding. An analysis of Hamming decoding will follow as a corollary. Concerning the loss L, our analysis assumes only that

$$\frac{L(z) + L(-z)}{2} \geq L(0) > 0 \tag{7}$$

for all $z \in \mathbb{R}$. Note that this property holds if L is convex, although convexity is by no means a necessary condition. Note also that all of the loss functions in Section 2 satisfy this property. The property is illustrated in Figure 1 for four of the loss functions discussed in that section.

Theorem 1 *Let ε be the average binary loss (as defined in Eq. (3)) of hypotheses $f_1, \ldots, f_\ell$ on a given training set $(x_1, y_1), \ldots, (x_m, y_m)$ with respect to the coding matrix $\mathbf{M} \in \{-1, 0, +1\}^{k \times \ell}$ and loss L, where k is the cardinality of the label set $\mathcal{Y}$. Let ρ be as in Eq. (6). Assume that L satisfies Eq. (7) for all $z \in \mathbb{R}$. Then the training error using loss-based decoding is at most*

$$\frac{\ell\varepsilon}{\rho L(0)}.$$

Proof: Suppose that loss-based decoding incorrectly classifies an example (x, y). Then there is some label $r \neq y$ for which

$$d_L(\mathbf{M}(r), \mathbf{f}(x)) \leq d_L(\mathbf{M}(y), \mathbf{f}(x)). \tag{8}$$

Let

$$S_\Delta = \{s : M(r, s) \neq M(y, s) \wedge M(r, s) \neq 0 \wedge M(y, s) \neq 0\}$$

be the set of columns of $\mathbf{M}$ in which rows r and y differ and are both non-zero. Let

$$S_0 = \{s : M(r, s) = 0 \vee M(y, s) = 0\}$$

be the set of columns in which either row r or row y is zero. Let $z_s = M(y, s) f_s(x)$ and $z'_s = M(r, s) f_s(x)$. Then Eq. (8) becomes

$$\sum_{s=1}^{\ell} L(z'_s) \leq \sum_{s=1}^{\ell} L(z_s)$$

which implies

$$\sum_{s \in S_\Delta \cup S_0} L(z'_s) \leq \sum_{s \in S_\Delta \cup S_0} L(z_s)$$

since $z_s = z'_s$ if $s \notin S_\Delta \cup S_0$. This in turn implies that

$$\begin{aligned}
\sum_{s=1}^{\ell} L(z_s) &\geq \sum_{s \in S_\Delta \cup S_0} L(z_s) \\
&\geq \frac{1}{2} \sum_{s \in S_\Delta \cup S_0} (L(z'_s) + L(z_s)) \\
&= \frac{1}{2} \sum_{s \in S_\Delta} (L(z'_s) + L(z_s)) \\
&\quad + \frac{1}{2} \sum_{s \in S_0} (L(z'_s) + L(z_s)). \qquad (9)
\end{aligned}$$

If $s \in S_\Delta$ then $z'_s = -z_s$ and, by assumption, $(L(-z_s) + L(z_s))/2 \geq L(0)$. Thus, the first term of Eq. (9) is at least $L(0)\ |S_\Delta|$. If $s \in S_0$, then either $z_s = 0$ or $z'_s = 0$. Either case implies that $L(z'_s) + L(z_s) \geq L(0)$. Thus, the second term of Eq. (9) is at last $L(0)\ |S_0|/2$.

Therefore, Eq. (9) is at least

$$L(0) \left(|S_\Delta| + \frac{|S_0|}{2} \right) = L(0) \Delta(\mathbf{M}(r), \mathbf{M}(y)) \geq \rho L(0).$$

In other words, a mistake on training example (x_i, y_i) implies that

$$\sum_{s=1}^{\ell} L(M(y_i, s) f_s(x_i)) \geq \rho L(0)$$

so the number of training mistakes is at most

$$\frac{1}{\rho L(0)} \sum_{i=1}^{m} \sum_{s=1}^{\ell} L(M(y_i, s) f_s(x_i)) = \frac{m \ell \varepsilon}{\rho L(0)}$$

and the training error is at most $\ell\varepsilon/(\rho L(0))$ as claimed. ■

As a corollary, we can give a similar but weaker theorem for Hamming decoding. Note that we use a different assumption about the loss function L, but one that also holds for all of the loss functions described in Section 2.

Corollary 2 *Let $f_1, \ldots, f_\ell$ be a set of hypotheses on a training set $(x_1, y_1), \ldots, (x_m, y_m)$, and let $\mathbf{M} \in \{-1, 0, +1\}^{k \times \ell}$ be a coding matrix where k is the cardinality of the label set $\mathcal{Y}$. Let ρ be as in Eq. (6). Then the training error using Hamming decoding is at most*

$$\frac{1}{\rho m} \sum_{i=1}^{m} \sum_{s=1}^{\ell} \left(1 - \mathrm{sign}(M(y_i, s) f_s(x_i))\right). \qquad (10)$$

Moreover, if L is a loss function satisfying $L(z) \geq L(0) > 0$ for $z < 0$ and ε is the average binary loss with respect to this loss function, then the training error using Hamming decoding is at most

$$\frac{2\ell\varepsilon}{\rho L(0)}. \qquad (11)$$

Proof: Consider the loss function $H(z) = (1 - \text{sign}(z))/2$. From Eqs. (4) and (5), it is clear that Hamming decoding is equivalent to loss-based decoding using this loss function. Moreover, H satisfies Eq. (7) for all z so we can apply Theorem 1 to get an upper bound on the training error of

$$\frac{2}{\rho m} \sum_{i=1}^{m} \sum_{s=1}^{\ell} H(M(y_i, s) f_s(x_i)) \tag{12}$$

which equals Eq. (10).

For the second part, note that if $z \leq 0$ then $H(z) \leq 1 \leq L(z)/L(0)$, and if $z > 0$ then $H(z) = 0 \leq L(z)/L(0)$. This implies that Eq. (12) is bounded above by Eq. (11). ■

Theorem 1 and Corollary 2 are broad generalizations of similar results proved by Schapire and Singer (1999) in a much more specialized setting involving only AdaBoost. Also, Corollary 2 generalizes some of the results of Guruswami and Sahai (1999) that bound the multiclass training error in terms of the training (misclassification) error rates of the binary classifiers.

The bounds of Theorem 1 and Corollary 2 depend implicitly on the fraction of zero entries in the matrix. Intuitively, the more zeros there are, the more examples that are ignored and the harder it should be to drive down the training error. At an extreme, if $\mathbf{M}$ is all zeros, then ρ is fairly large ($\ell/2$) but learning certainly should not be possible. To make this dependence explicit, let

$$T = \{(i, s) : M(y_i, s) = 0\}$$

be the set of pairs i, s inducing examples that are ignored during learning. Let $q = |T|/(m\ell)$ be the fraction of ignored pairs. Let ε be the average binary loss *restricted* to the pairs not ignored during training:

$$\varepsilon = \frac{1}{|T^c|} \sum_{(i,s) \in T^c} L(M(y_i, s) f_s(x_i))$$

where $T^c = \{(i, s) : M(y_i, s) \neq 0\}$. Then the bound in Theorem 1 can be rewritten

$$\frac{\ell}{\rho L(0)} \frac{1}{m\ell} \left(\sum_{(i,s) \in T} L(0) + \sum_{(i,s) \notin T} L(M(y_i, s) f_s(x_i)) \right) = \frac{\ell}{\rho} \left(q + (1 - q) \frac{\varepsilon}{L(0)} \right).$$

Similarly, let ϵ be the fraction of misclassification errors made on T^c:

$$\epsilon = \frac{1}{|T^c|} \sum_{(i,s) \in T^c} [\![M(y_i, s) \neq \text{sign}(f_s(x_i))]\!].$$

The first part of Corollary 2 implies that the training error using Hamming decoding is bounded above by

$$\frac{\ell}{\rho}(q + 2(1 - q)\epsilon).$$

We see from these bounds that there are many trade-offs in the design of the coding matrix $\mathbf{M}$. On the one hand, we want the rows to be far apart so that ρ will be large, and we also want there to be few non-zero entries so that q will be small. On the other hand, attempting to make ρ large and q small may produce binary problems that are difficult to learn, yielding large (restricted) average binary loss.

5. Analysis of Generalization Error for Boosting with Loss-based Decoding

The previous section considered only the training error using output codes. In this section, we take up the more difficult task of analyzing the generalization error. Because of the difficulty of obtaining such results, we do not have the kind of general results obtained for training error which apply to a broad class of loss functions. Instead, we focus only on the generalization error of using AdaBoost with output coding and loss-based decoding. Specifically, we show how the margin-theoretic analysis of Schapire et al. (1998) can be extended to this more complicated algorithm.

Briefly, Schapire et al.'s analysis was proposed as a means of explaining the empirically observed tendency of AdaBoost to resist overfitting. Their theory was based on the notion of an example's *margin* which, informally, measures the "confidence" in the prediction made by a classifier on that example. They then gave a two-part analysis of AdaBoost: First, they proved a bound on the generalization error in terms of the margins of the training examples, a bound that is independent of the number of base hypotheses combined, and a bound suggesting that larger margins imply lower generalization error. In the second part of their analysis, they proved that AdaBoost tends to aggressively increase the margins of the training examples.

In this section, we give counterparts of these two parts of their analysis for the combination of AdaBoost with loss-based decoding. We also assume that the single-call variant is used as described in Section 3. The result is essentially the AdaBoost.MO algorithm of Schapire and Singer (1999) (specifically, what they called "Variant 2").

This algorithm works as follows. We assume that a coding matrix $\mathbf{M}$ is given. The algorithm works in rounds, repeatedly calling the base learning algorithm to obtain a base hypothesis. On each round $t = 1, \ldots, T$, the algorithm computes a distribution D_t over pairs of training examples and columns of the matrix $\mathbf{M}$, i.e., over the set $\{1, \ldots, m\} \times \mathcal{L}$ where $\mathcal{L} = \{1, \ldots, \ell\}$. The base learning algorithm uses the training data (with binary labels as encoded using $\mathbf{M}$) and the distribution D_t to obtain a base hypothesis $h_t : \mathcal{X} \times \mathcal{L} \to \{-1, +1\}$. (In general, h_t's range may be $\mathbb{R}$, but here, for simplicity, we assume that h_t is binary valued.) The error ϵ_t of h_t is the probability with respect to D_t of misclassifying one of the examples. That is,

$$\begin{aligned} \epsilon_t &= \Pr_{(i,s)\sim D_t}\left[M(y_i, s) \neq h_t(x_i, s)\right] \\ &= \sum_{i=1}^{m} \sum_{s=1}^{\ell} D_t(i,s) [\![M(y_i, s) \neq h_t(x_i, s)]\!]. \end{aligned}$$

The distribution D_t is then updated using the rule

$$D_{t+1}(i,s) = \frac{D_t(i,s) \exp(-\alpha_t M(y_i, s) h_t(x_i, s))}{Z_t}. \tag{13}$$

Here, $\alpha_t = (1/2) \ln((1 - \epsilon_t)/\epsilon_t)$ (which is nonnegative, assuming, as we do, that $\epsilon_t \leq 1/2$), and Z_t is a normalization constant ensuring that D_{t+1} is a distribution. It is straightforward to show that

$$Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)}. \tag{14}$$

(The initial distribution is chosen to be uniform so that $D_1(i,s) = 1/(m\ell)$.)

After T rounds, this procedure outputs a final classifier H which, because we are using loss-based decoding, is

$$H(x) = \arg\min_{y \in \mathcal{Y}} \sum_{s=1}^{\ell} \exp\left(-M(y,s) \sum_{t=1}^{T} \alpha_t h_t(x,s)\right). \tag{15}$$

We begin our margin-theoretic analysis with a definition of the margin of this combined multiclass classifier. First, let

$$\nu(f, \eta, x, y) = -\frac{1}{\eta} \ln \left(\frac{1}{\ell} \sum_{s=1}^{\ell} e^{-\eta M(y,s) f(x,s)} \right).$$

If we let

$$\eta = \sum_{t=1}^{T} \alpha_t, \tag{16}$$

and

$$f(x, s) = \frac{1}{\eta} \sum_{t=1}^{T} \alpha_t h_t(x, s), \tag{17}$$

we can then rewrite Eq. (15) as

$$H(x) = \arg\max_{y \in \mathcal{Y}} \nu(f, \eta, x, y). \tag{18}$$

Since we have transformed the argument of the minimum in Eq. (15) by a strictly decreasing function (namely, $x \mapsto -(1/\eta)\ln(x/\ell)$) to arrive at Eq. (18) it is clear that we have not changed the definition of H. This rewriting has the effect of normalizing the argument of the maximum in Eq. (18) so that it is always in the range $[-1, +1]$. We can now define the *margin* for a labeled example (x, y) to be the difference between the vote $\nu(f, \eta, x, y)$ given to the correct label y, and the largest vote given to any other label. We denote the margin by $\mathcal{M}_{f,\eta}(x, y)$. Formally,

$$\mathcal{M}_{f,\eta}(x, y) = \frac{1}{2} \left[\nu(f, \eta, x, y) - \max_{r \neq y} \nu(f, \eta, x, r) \right],$$

where the factor of $1/2$ simply ensures that the margin is in the range $[-1, +1]$. Note that the margin is positive if and only if H correctly classifies example (x, y).

Although this definition of margin is seemingly very different from the one given earlier in the paper for binary problems (which is the same as the one used by Schapire et al. in their comparatively simple context), we show next that maximizing training-example margins translates into a better bound on generalization error, independent of the number of rounds of boosting.

Let $\mathcal{H}$ be the base-hypothesis space of $\{-1, +1\}$-valued functions on $\mathcal{X} \times \mathcal{L}$. We let $\mathrm{co}(\mathcal{H})$ denote the *convex hull* of $\mathcal{H}$:

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

where it is understood that each of the sums above is over the finite subset of hypotheses in $\mathcal{H}$ for which $\alpha_h > 0$. Thus, f as defined in Eq. (17) belongs to $\mathrm{co}(\mathcal{H})$.

We assume that training examples are chosen i.i.d. from some distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$. We write probability or expectation with respect to a random choice of an example (x, y) according to $\mathcal{D}$ as $\Pr_{\mathcal{D}}[\cdot]$ and $\mathrm{E}_{\mathcal{D}}[\cdot]$. Similarly, probability and expectation with respect to an example chosen uniformly at random from training set S is denoted $\Pr_S[\cdot]$ and $\mathrm{E}_S[\cdot]$.

We can now prove the first main theorem of this section which shows how the generalization error can be usefully bounded when most of the training examples have large margin. This is very similar to the results of Schapire et al. (1998) except for the fact that it applies to loss-based decoding for a general coding matrix $\mathbf{M}$.

Theorem 3 *Let $\mathcal{D}$ be a distribution over $X \times \mathcal{Y}$, and let S be a sample of m examples chosen independently at random according to $\mathcal{D}$. Suppose the base-classifier space $\mathcal{H}$ has VC-dimension d, and let $\delta > 0$. Assume that $m \geq d\ell \geq 1$ where ℓ is the number of columns in the coding matrix $\mathbf{M}$. Then with probability at least $1-\delta$ over the random choice of the training set S, every weighted average function $f \in \mathrm{co}(\mathcal{H})$ and every $\eta > 0$ satisfies the following bound for all $\theta > 0$:*

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

Proof: To prove the theorem, we will first need to define the notion of a sloppy cover, slightly specialized for our purposes. For a class $\mathcal{F}$ of real-valued functions over $\mathcal{X} \times \mathcal{Y}$, a training set $S \subset \mathcal{X} \times \mathcal{Y}$ 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 $\mathrm{Pr}_S\left[|\hat{F}(x,y) - F(x,y)| > \theta\right] \leq \epsilon$. 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.

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 $F \in \mathcal{F}$ for which

$$\mathrm{Pr}_{\mathcal{D}}\left[F(x,y) \leq 0\right] > \mathrm{Pr}_S\left[F(x,y) \leq \theta\right] + \epsilon$$

is at most

$$2\mathcal{N}(\mathcal{F},\theta/2,\epsilon/8,2m)e^{-\epsilon^2 m/32}. \tag{19}$$

We prove Theorem 3 by applying this result to the family of functions

$$\mathcal{F} = \{\mathcal{M}_{f,\eta} : f \in \mathrm{co}(\mathcal{H}), \eta > 0\}.$$

To do so, we need to construct a relatively small set of functions that approximate all the functions in $\mathcal{F}$.

We start with a lemma that implies that any function $\mathcal{M}_{f,\eta}$ can be approximated by $\mathcal{M}_{f,\hat{\eta}}$ for some $\hat{\eta}$ in the small finite set

$$\mathcal{E}_\theta = \left\{\frac{\ln \ell}{i\theta} : i = 1, \ldots, \left\lceil \frac{\ln \ell}{2\theta^2} \right\rceil \right\}.$$

Lemma 4 *For all $\eta > 0$, there exists $\hat{\eta} \in \mathcal{E}_\theta$ such that for all $f \in \mathrm{co}(\mathcal{H})$ and for all $x \in \mathcal{X}$, $r \in \mathcal{Y}$,*

$$|\nu(f,\eta,x,y) - \nu(f,\hat{\eta},x,y)| \leq \theta.$$

Proof: Let

$$\Lambda(\eta, \mathbf{z}) = \frac{1}{\eta} \ln \left(\frac{1}{\ell} \sum_{s=1}^{\ell} e^{\eta z_s} \right)$$

for $\mathbf{z} \in \mathbb{R}^\ell$. We claim first that, for any $\mathbf{z}$, and for $0 < \eta_1 \le \eta_2$,

$$0 \le \Lambda(\eta_2, \mathbf{z}) - \Lambda(\eta_1, \mathbf{z}) \le \left(\frac{1}{\eta_1} - \frac{1}{\eta_2} \right) \ln \ell. \tag{20}$$

For the first inequality, it suffices to show that $F(\eta) = \Lambda(\eta, \mathbf{z})$ is nondecreasing. Differentiating, we find that

$$\frac{dF}{d\eta} = \frac{\ln \ell + \sum_{s=1}^{\ell} p_s \ln p_s}{\eta^2} \tag{21}$$

where $p_s = e^{\eta z_s} / \sum_{s=1}^{\ell} e^{\eta z_s}$. Since entropy over ℓ symbols cannot exceed $\ln \ell$, this quantity is nonnegative.

For the second inequality of Eq. (20), it suffices to show that $G(\eta) = \Lambda(\eta, \mathbf{z}) + (\ln \ell)/\eta$ is nonincreasing. Again differentiating (or reusing Eq. (21)), we find that

$$\frac{dG}{d\eta} = \frac{\sum_{s=1}^{\ell} p_s \ln p_s}{\eta^2}$$

which is nonpositive since entropy cannot be negative.

So, if $\eta \ge \min \mathcal{E}_\theta$, then let $\hat{\eta} = (\ln \ell)/(i\theta)$ be the largest element of $\mathcal{E}_\theta$ that is no bigger than η. If $i > 1$ then

$$\frac{\ln \ell}{i\theta} \le \eta < \frac{\ln \ell}{(i-1)\theta}$$

so

$$\begin{aligned} 0 \le \Lambda(\eta, \mathbf{z}) - \Lambda(\hat{\eta}, \mathbf{z}) &\le \left(\frac{i\theta}{\ln \ell} - \frac{1}{\eta} \right) \ln \ell \\ &\le \left(\frac{i\theta}{\ln \ell} - \frac{(i-1)\theta}{\ln \ell} \right) \ln \ell = \theta. \end{aligned}$$

If $i = 1$, then

$$0 \le \Lambda(\eta, \mathbf{z}) - \Lambda(\hat{\eta}, \mathbf{z}) \le \left(\frac{\theta}{\ln \ell} - \frac{1}{\eta} \right) \ln \ell \le \theta.$$

It remains then only to handle the case that η is small. Assume that $\mathbf{z} \in [-1, +1]^\ell$. Then

$$\begin{aligned} \Lambda(\eta, \mathbf{z}) &= \frac{1}{\eta} \ln \left(\frac{1}{\ell} \sum_{s=1}^{\ell} e^{\eta z_s} \right) \\ &\le \frac{1}{\eta} \ln \left(\exp \left(\frac{\eta^2}{2} + \frac{\eta}{\ell} \sum_{s=1}^{\ell} z_s \right) \right) \\ &= \frac{\eta}{2} + \frac{1}{\ell} \sum_{s=1}^{\ell} z_s. \end{aligned}$$

This is because, as proved by Hoeffding (1963), for any random variable X with $a \leq X \leq b$, and for $\eta > 0$,

$$\mathrm{E}\left[e^{\eta X}\right] \leq \exp\left(\frac{\eta^2(b-a)^2}{8} + \eta \mathrm{E}\left[X\right]\right).$$

On the other hand, by Eq. (20),

$$\begin{aligned}
\Lambda(\eta, \mathbf{z}) &\geq \lim_{\eta \to 0} \Lambda(\eta, \mathbf{z}) \\
&= \lim_{\eta \to 0} \frac{\frac{1}{\ell}\sum_{s=1}^{\ell} e^{\eta z_s} z_s}{\frac{1}{\ell}\sum_{s=1}^{\ell} e^{\eta z_s}} \\
&= \frac{1}{\ell}\sum_{s=1}^{\ell} z_s
\end{aligned}$$

where the first equality uses l'Hôpital's rule. Thus, if $\eta < \min \mathcal{E}_\theta$, then we take $\hat{\eta} = \min \mathcal{E}_\theta \approx 2\theta$ so that

$$\frac{1}{\ell}\sum_{s=1}^{\ell} z_s \leq \Lambda(\eta, \mathbf{z}) \leq \Lambda(\hat{\eta}, \mathbf{z}) \leq \frac{1}{\ell}\sum_{s=1}^{\ell} z_s + \frac{\hat{\eta}}{2}$$

which implies that

$$|\Lambda(\eta, \mathbf{z}) - \Lambda(\hat{\eta}, \mathbf{z})| \leq \frac{\hat{\eta}}{2} \leq \theta$$

assuming $\mathbf{z} \in [-1, +1]^\ell$. Since $\nu(f, \eta, x, r) = -\Lambda(\eta, \mathbf{z})$ with $z_s = -M(r, s) f(x, s) \in [-1, +1]$, this completes the lemma. ■

Let S be a fixed subset of $\mathcal{X} \times \mathcal{Y}$ of size m. Because $\mathcal{H}$ has VC-dimension d, there exists a subset $\hat{\mathcal{H}}$ of $\mathcal{H}$ of cardinality $(e\ell m/d)^d$ that includes all behaviors on S. That is, for all $h \in \mathcal{H}$, there exists $\hat{h} \in \hat{\mathcal{H}}$ such that $h(x, s) = \hat{h}(x, s)$ for all $(x, y) \in S$ and all $s \in \mathcal{L}$. Now let

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

be the set of unweighted averages of N elements in $\hat{\mathcal{H}}$, and let

$$\hat{\mathcal{F}}_{N,\theta} = \left\{ \mathcal{M}_{f,\eta} : f \in \mathcal{C}_N, \eta \in \mathcal{E}_\theta \right\}.$$

We will show that $\hat{\mathcal{F}}_{N,\theta}$ is a sloppy cover of $\mathcal{F}$.

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

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

where $\alpha_j \geq 0$ and $\sum_j \alpha_j = 1$. Because we are only interested in the behavior of f on points in S, we can assume without loss of generality that each $h_j \in \hat{\mathcal{H}}$.

Lemma 5 *Suppose for some $x \in \mathcal{X}$ and some $g \in \mathcal{C}_N$, we have that $|f(x, s) - g(x, s)| \leq \theta$ for all $s \in \mathcal{L}$. Let $\eta > 0$ and let $\hat{\eta} \in \mathcal{E}_\theta$ be as in Lemma 4. Then for all $y \in \mathcal{Y}$,*

$$|\mathcal{M}_{f,\eta}(x, y) - \mathcal{M}_{g,\hat{\eta}}(x, y)| \leq 2\theta.$$

Proof: For all $r \in \mathcal{Y}$,

$$\begin{aligned}
\nu(f,\eta,x,r) - \nu(g,\eta,x,r) &= \frac{1}{\eta}\ln\left(\frac{\sum_s \exp\left(-\eta M(r,s)g(x,s)\right)}{\sum_s \exp\left(-\eta M(r,s)f(x,s)\right)}\right) \\
&\le \frac{1}{\eta}\ln\left(\max_s \exp\left(-\eta M(r,s)(g(x,s)-f(x,s))\right)\right) \\
&= \max_s M(r,s)(f(x,s)-g(x,s)) \\
&\le \max_s |M(r,s)||f(x,s)-g(x,s)| \le \theta
\end{aligned}$$

where the first inequality uses the simple fact that $(\sum_i a_i)/(\sum_i b_i) \le \max_i a_i/b_i$ for positive a_i's and b_i's. By the symmetry of this argument,

$$|\nu(f,\eta,x,r) - \nu(g,\eta,x,r)| \le \theta.$$

Also, from Lemma 4,

$$|\nu(g,\eta,x,r) - \nu(g,\hat{\eta},x,r)| \le \theta$$

so

$$|\nu(f,\eta,x,r) - \nu(g,\hat{\eta},x,r)| \le 2\theta$$

for all $r \in \mathcal{Y}$. By definition of margin, this implies the lemma. ■

Recall that the coefficients α_j are nonnegative and that they sum to one; in other words, they define a probability distribution over $\hat{\mathcal{H}}$. It will be useful to imagine sampling from this distribution. Specifically, suppose that $\hat{h} \in \hat{\mathcal{H}}$ is chosen in this manner. Then for fixed (x,s), $\hat{h}(x,s)$ is a $\{-1,+1\}$-valued random variable with expected value of exactly $f(x,s)$. Consider now choosing N such functions $\hat{h}_1,\ldots,\hat{h}_N$ independently at random, each according to this same distribution, and let $g = (1/N)\sum_{i=1}^N \hat{h}_i$. Then $g \in \mathcal{C}_N$. Let us denote by $\mathcal{Q}$ the resulting distribution over functions in $\mathcal{C}_N$. Note that, for fixed (x,s), $g(x,s)$ is the average of N $\{-1,+1\}$-trials with expected value $f(x,s)$.

For any $(x,y) \in S$,

$$\begin{aligned}
\mathrm{Pr}_{g\sim\mathcal{Q}}\left[|\mathcal{M}_{f,\eta}(x,y) - \mathcal{M}_{g,\hat{\eta}}(x,y)| > 2\theta\right] &\le \mathrm{Pr}_{g\sim\mathcal{Q}}\left[\exists s \in \mathcal{L} : |f(x,s)-g(x,s)| > \theta\right] \\
&\le \sum_{s=1}^{\ell} \mathrm{Pr}_{g\sim\mathcal{Q}}\left[|f(x,s)-g(x,s)| > \theta\right] \\
&\le 2\ell e^{-N\theta^2/2}.
\end{aligned}$$

These three inequalities follow, respectively, from Lemma 5, the union bound and Hoeffding's inequality. Thus,

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

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

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

We have thus shown that $\hat{\mathcal{F}}_{N,\theta}$ is a $2\ell e^{-N\theta^2/2}$-sloppy 2θ-cover of $\mathcal{F}$. In other words,

$$\mathcal{N}(\mathcal{F}, 2\theta, 2\ell e^{-N\theta^2/2}, m) \leq |\hat{\mathcal{F}}_{N,\theta}| \leq \left(\frac{e\ell m}{d}\right)^{dN} \left\lceil \frac{\ln \ell}{2\theta^2} \right\rceil .$$

Making the appropriate substitutions, this gives that Eq. (19) is at most

$$\frac{16 \ln \ell}{\theta^2} \left(\frac{2e\ell m}{d}\right)^{(32d/\theta^2)\ln(16\ell/\epsilon)} e^{-m\epsilon^2/32}. \tag{22}$$

Let

$$\epsilon = 16 \left(\frac{\ln\left(\frac{16 \ln l}{\delta\theta^2}\right)}{8m} + \frac{2d}{m\theta^2} \ln\left(\frac{2e\ell m}{d}\right) \ln\left(\frac{e\ell^2 m}{d}\right) \right)^{1/2} .$$

Note that the quantity inside the square root is at least $2d/(m\theta^2) \geq d/(me)$. Thus, $\epsilon \geq 16\sqrt{d/(me)}$. Using this approximation for the first occurrence of ϵ, it follows that Eq. (22) is at most δ.

We have thus proved the bound of the theorem for a single given choice of $\theta > 0$ with high probability. To prove that the bound holds simultaneously for all $\theta > 0$, we can use exactly the same argument used by Schapire and Singer (1999) in the very last part of their Theorem 8. ■

Thus, we have shown that large training-set margins imply a better bound on the generalization error, independent of the number of rounds of boosting. We turn now to the second part of our analysis in which we prove that AdaBoost.MO tends to increase the margins of the training examples, assuming that the binary errors ϵ_t of the base hypotheses are bounded away from the trivial error rate of $1/2$ (see the discussion that follows the proof). The theorem that we prove below is a direct analog of Theorem 5 of Schapire et al. (1998) for binary AdaBoost. Note that we focus only on coding matrices that do not contain zeros. A slightly weaker result can be proved in the more general case.

Theorem 6 *Suppose the base learning algorithm, when called by AdaBoost.MO using coding matrix $\mathbf{M} \in \{-1,+1\}^{k\times\ell}$, generates hypotheses with weighted binary training errors $\epsilon_1, \ldots, \epsilon_T$. Let ρ be as in Eq. (6). Then for any $\theta \geq 0$, we have that*

$$\mathrm{Pr}_S\left[\mathcal{M}_{f,\eta}(x,y) \leq \theta\right] \leq \frac{\ell}{\rho} \prod_{t=1}^{T} \left[2\sqrt{\epsilon_t^{1-\theta}(1-\epsilon_t)^{1+\theta}}\right]$$

where f and η are as in Eqs. (16) and (17).

Proof: Suppose, for some labeled example (x,y), $\mathcal{M}_{f,\eta}(x,y) \leq \theta$. Then, by definition of margin, there exists a label $r \in \mathcal{Y} - \{y\}$ for which

$$\frac{1}{2}(\nu(f,\eta,x,y) - \nu(f,\eta,x,r)) \leq \theta,$$

that is

$$\sum_{s=1}^{\ell} e^{-\eta M(r,s) f(x,s)} \leq e^{2\eta\theta} \sum_{s=1}^{\ell} e^{-\eta M(y,s) f(x,s)}. \tag{23}$$

Let $z_s = \eta M(y,s) f(x,s) - \eta\theta$ and $z'_s = \eta M(r,s) f(x,s) + \eta\theta$. Let $S_\Delta = \{s \in \mathcal{L} : M(r,s) \neq M(y,s)\}$. Then Eq. (23) implies

$$\sum_{s=1}^{\ell} e^{-z_s} \geq \sum_{s=1}^{\ell} e^{-z'_s}$$

and so

$$\begin{aligned} \sum_{s=1}^{\ell} e^{-z_s} &\geq \sum_{s=1}^{\ell} \frac{e^{-z_s} + e^{-z'_s}}{2} \\ &\geq \sum_{s \in S_\Delta} \frac{e^{-z_s} + e^{-z'_s}}{2} \\ &= \sum_{s \in S_\Delta} \frac{e^{-z_s} + e^{z_s}}{2} \geq |S_\Delta| \geq \rho. \end{aligned}$$

This is because, if $s \in S_\Delta$ then $z'_s = -z_s$, and because $x + 1/x \geq 2$ for all $x > 0$.

Therefore, if $\mathcal{M}_{f,\eta}(x,y) \leq \theta$ then

$$\sum_{s=1}^{\ell} e^{-\eta M(y,s) f(x,s)} \geq \rho e^{-\eta\theta}.$$

Thus, the fraction of training examples i for which $\mathcal{M}_{f,\eta}(x_i, y_i) \leq \theta$ is at most

$$\begin{aligned} &\frac{e^{\eta\theta}}{\rho m} \sum_{i=1}^{m} \sum_{s=1}^{\ell} e^{-\eta M(y_i,s) f(x_i,s)} \\ &= \frac{\ell e^{\eta\theta}}{\rho} \sum_{i=1}^{m} \sum_{s=1}^{\ell} \frac{1}{m\ell} \exp\left(-\sum_{t=1}^{T} \alpha_t M(y_i,s) h_t(x_i,s)\right) &(24) \\ &= \frac{\ell e^{\eta\theta}}{\rho} \sum_{i=1}^{m} \sum_{s=1}^{\ell} \left(\prod_{t=1}^{T} Z_t\right) D_{T+1}(i,s) &(25) \\ &= \frac{\ell e^{\eta\theta}}{\rho} \left(\prod_{t=1}^{T} Z_t\right). &(26) \end{aligned}$$

Here, Eq. (24) uses the definition of f and η as in Eqs. (16) and (17). Eq. (25) uses the definition of D_{T+1} as defined recursively in Eq. (13). Eq. (26) uses the fact that D_{T+1} is a distribution. The theorem now follows by plugging in Eq. (14) and applying straightforward algebra. ■

As noted by Schapire et al. (1998), this bound can be usefully understood if we assume that $\epsilon_t \leq 1/2 - \gamma$ for all t. Then the upper bound simplifies to

$$\frac{\ell}{\rho}\left(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}}\right)^{T}.$$

If $\theta < \gamma$, then the expression inside the parentheses is smaller than one so that the fraction of training examples with margin below θ drops to zero exponentially fast in T.

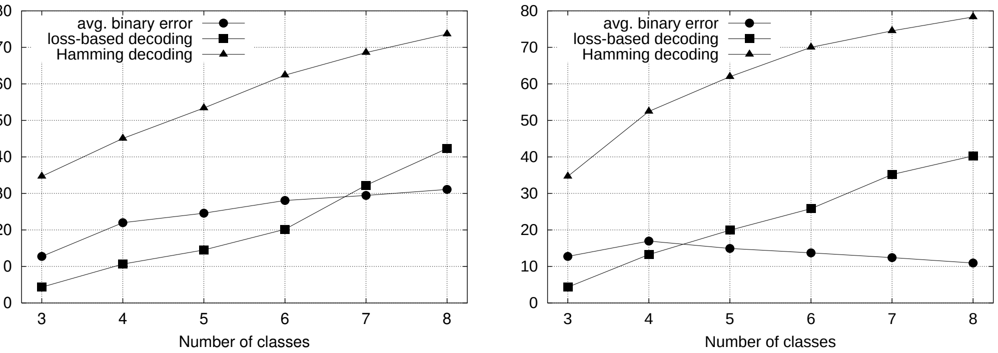

Figure 3: Comparison of the average binary error, the multiclass error using Hamming decoding, and the multiclass error using loss-based decoding with AdaBoost on synthetic data, using a complete code (left) and the one-against-all code (right).

6. Experiments

In this section, we describe and discuss experiments we have performed with synthetic data and with natural data from the UCI repository. We run both sets of experiments with two base-learners: AdaBoost and SVM. Two primary goals of the experiments are to compare Hamming decoding to loss-based decoding and to compare the performance of different output codes. We start with a description of an experiment with real-valued synthetic data which underscores the tradeoff between the complexity of the binary problems induced by a given output code and its error correcting properties.

In the experiments with synthetic data, we generated instances according to the normal distribution with zero mean and a unit variance. To create a multiclass problem with k classes, we set $k+1$ thresholds, denoted $\theta_0, \theta_1, \ldots, \theta_k$, where $\theta_0 = 0$ and $\theta_k = \infty$. An instance x is associated with class j if and only if $\theta_{j-1} \leq |x| < \theta_j$. For each $k \in \{3, \ldots, 8\}$ we generated 10 sets of examples, where each set was of size $100k$. The thresholds were set so that exactly 100 examples will be associated with each class. Similarly, we generated the same number of test sets (of the same size) using the thresholds obtained for the training data to label the test data. We used AdaBoost as the base learner and set the number of rounds of boosting for each binary problem to 10 and called AdaBoost repeatedly for each column (multi-call variant). For weak hypotheses, we used the set of all possible threshold functions. That is, a weak hypothesis based on a threshold t would label an instance x as $+1$ if $|x| < t$ and -1 otherwise.

In Figure 3, we plot the test error rate as the function of k for two output coding matrices. The first is a complete code whose columns correspond to all the non-trivial partitions of the set $\{1, \ldots, k\}$ into two subsets. Thus, this code is a matrix of size $k \times (2^{k-1} - 1)$. The second code was the one-against-all code. For each code, we plot the average binary test error, the multiclass errors using Hamming decoding, and the multiclass errors using loss-based decoding. The graphs clearly show that Hamming decoding is inferior to loss-based decoding and yields much higher error rates. The multiclass errors of the two codes using loss-based decoding are comparable. While the multiclass error rate with the complete code is slightly lower than the error rate for the one-against-all code, the situation is reversed for the average binary errors. This phenomenon underscores the

Problem	#Examples Train	Test	#Attributes	#Classes
dermatology	366	-	34	6
satimage	4435	2000	36	6
glass	214	-	9	7
segmentation	2310	-	19	7
ecoli	336	-	8	8
pendigits	7494	3498	16	10
yeast	1484	-	8	10
vowel	528	462	10	11
soybean	307	376	35	19
thyroid	9172	-	29	20
audiology	226	-	69	24
isolet	6238	1559	617	26
letter	16000	4000	16	26

Table 1: Description of the datasets used in the experiments.

tradeoff between the redundancy and correcting properties of the output codes and the difficulty of the binary learning problems it induces. The complete code has good error correcting properties; the distance between each pair of rows is $\rho = (l+1)/2 = 2^{k-2}$. However, many of the binary problems that the complete code induces are difficult to learn. The distance between each pair of rows in the one-against-all code is small: $\rho = 2$. Hence, its empirical error bound according to Theorem 1 seems inferior. However, the binary problems it induces are simpler and thus its average binary loss ε is lower than the average binary loss of the complete code and the overall result is comparable performance.

Next, we describe experiments we performed with multiclass data from the UCI repository (Merz & Murphy, 1998). We used two different popular binary learners, AdaBoost and SVM. We chose the following datasets (ordered according to the number of classes): Dermatology, Satimage, Glass, Segmentation, E-coli, Pendigits, Yeast, Vowel, Soybean-large, Thyroid, Audiology, Isolet, Letter-recognition. The properties of the datasets are summarized in Table 1. In the SVM experiments, we skipped Audiology, Isolet, Letter-recognition, Segmentation, and Thyroid, because these datasets were either too big to be handled by our current implementation of SVM or contained many nominal features with missing values which are problematic for SVM. All datasets have at least six classes. When available, we used the original partition of the datasets into training and test sets. For the other datasets, we used 10-fold cross validation. For SVM, we used polynomial kernels of degree 4 with the multi-call variant. For AdaBoost, we used decision stumps for base hypotheses. By modifying an existing package of AdaBoost.MH (Schapire & Singer, 1999) we were able to devise a simple implementation of the single-call variant that was described in Section 5. Summaries of the results using the different output codes described below are given in Tables 2 and 3.

We tested five different types of output codes: one-against-all, complete (in which there is one column for every possible (non-trivial) split of the classes), all-pairs, and two types of random codes. The first type of random code has $\lceil 10 \log_2(k) \rceil$ columns for a problem with k classes. Each

Problem	One-vs-all	Complete	All-Pairs	Dense	Sparse
	Hamming Decoding				
dermatology	5.0	4.2	3.1	3.9	3.6
satimage	14.9	12.3	11.7	12.3	13.2
glass	31.0	31.0	28.6	28.6	27.1
segmentation	0.0	0.1	0.0	0.1	0.1
ecoli	21.5	18.5	19.1	17.6	19.7
pendigits	8.9	8.6	3.0	9.3	6.2
yeast	44.7	41.9	42.5	43.9	49.5
vowel	67.3	59.3	50.2	62.6	54.5
soybean	8.2	–	9.0	5.6	8.0
thyroid	7.8	–	–	12.3	8.1
audiology	26.9	–	23.1	23.1	23.1
isolet	9.2	–	–	10.8	10.1
letter	27.7	–	7.8	30.9	27.1
	Loss-based Decoding (L_1)				
dermatology	4.2	4.2	3.1	3.9	3.6
satimage	12.1	12.4	11.2	11.9	11.9
glass	26.7	31.0	27.1	27.1	26.2
segmentation	0.0	0.1	0.0	0.1	0.7
ecoli	17.3	17.6	18.8	18.5	19.1
pendigits	4.6	8.6	2.9	8.8	6.8
yeast	41.6	42.0	42.6	43.2	49.8
vowel	56.9	59.1	50.9	61.9	54.1
soybean	7.2	–	8.8	4.8	8.2
thyroid	6.5	–	–	12.0	8.0
audiology	19.2	–	23.1	19.2	23.1
isolet	5.3	–	–	10.1	9.8
letter	14.6	–	7.4	29.0	26.6
	Loss-based Decoding (Exp.)				
dermatology	4.2	3.9	3.1	4.2	3.1
satimage	12.1	12.3	11.4	12.0	12.0
glass	26.7	28.6	27.6	25.2	29.0
segmentation	0.0	0.0	0.0	0.0	0.0
ecoli	15.8	16.4	18.2	17.0	17.9
pendigits	4.6	7.2	2.9	8.1	4.8
yeast	41.6	42.1	42.3	43.0	49.3
vowel	56.9	54.1	51.7	60.0	49.8
soybean	7.2	–	8.8	4.8	5.6
thyroid	6.5	–	–	11.4	7.2
audiology	19.2	–	23.1	19.2	19.2
isolet	5.3	–	–	9.4	9.7
letter	14.6	–	7.1	28.3	22.3

Table 2: Results of experiments with output codes with datasets from the UCI repository using AdaBoost as the base binary learner. For each problem five output codes were used and then evaluated (see text) with three decoding methods: Hamming decoding, loss-based decoding using AdaBoost with randomized predictions (denoted L_1), and loss-based decoding using the exponential loss function (denoted *Exp*).

	Hamming Decoding				
Problem	One-vs-all	Complete	All-Pairs	Dense	Sparse
dermatology	4.2	3.6	3.1	3.6	2.5
satimage	40.9	14.3	50.4	15.0	27.4
glass	37.6	34.3	29.5	34.8	32.4
ecoli	15.8	14.2	13.9	15.2	14.2
pendigits	3.9	2.0	26.2	2.5	2.6
yeast	73.9	42.4	40.8	42.5	48.1
vowel	60.4	53.0	39.2	53.5	50.2
soybean	20.5	–	9.6	9.0	9.0
	Loss-based Decoding				
dermatology	3.3	3.6	3.6	3.9	3.1
satimage	40.9	13.9	27.8	14.3	13.3
glass	38.6	34.8	31.0	34.8	32.4
ecoli	16.1	13.6	13.3	14.8	14.8
pendigits	2.5	1.9	3.1	2.1	2.7
yeast	72.9	40.5	40.9	39.7	47.2
vowel	50.9	51.3	39.0	51.7	47.0
soybean	21.0	–	10.4	8.8	9.0

Table 3: Results of experiments with output codes with datasets from the UCI repository using the support-vector machine (SVM) algorithm as the base binary learner. For each problem five different classes of output codes were tested were used and then evaluated with Hamming decoding and the appropriate loss-based decoding for SVM.

element in the code was chosen uniformly at random from $\{-1, +1\}$. For brevity, we call these dense random codes. We generated a dense random code for each multiclass problem by examining 10,000 random codes and choosing the code that had the largest ρ and did not have any identical columns. The second type of code, called a sparse code, was chosen at random from $\{-1, 0, +1\}$. Each element in a sparse code is 0 with probability $\frac{1}{2}$ and -1 or $+1$ with probability $\frac{1}{4}$ each. The sparse codes have $\lceil 15 \log_2(k) \rceil$ columns. For each problem, we picked a code with high value of ρ by examining 10,000 random codes as before. However, now we also had to check that no code had a column or a row containing only zeros. Note that for some of the problems with many classes, we could not evaluate the complete and all-pairs codes since they were too large.

We compared Hamming decoding to loss-based decoding for each of the five families of codes. The results are plotted in Figures 4 and 5. Each of the tested UCI datasets is plotted as a bar in the figures where height of the bar (possibly negative) is proportional to the test error rate of loss-based decoding minus the test error rate of Hamming decoding. The datasets are indexed $1, 2, \ldots$ and are plotted in the order listed above. We tested AdaBoost with two loss functions for decoding: the exponential loss (denoted "Exp" in the figure and drawn in black) and the loss function $1/(1 + e^{2yf(x)})$ (denoted "L_1" and drawn in gray) which is the result of using AdaBoost

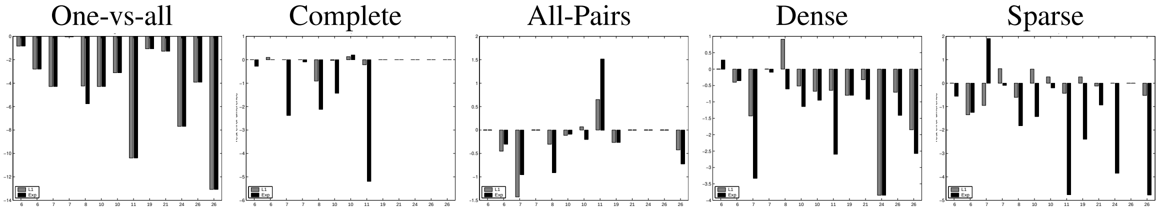

Figure 4: Comparison of the test error using Hamming decoding and loss-based decoding when the binary learners are trained using AdaBoost. Two loss functions for decoding are plotted: the exponential loss ("Exp", in black) and $1/(1+e^{2yf(x)})$ when using AdaBoost with randomized predictions ("L_1", in gray).

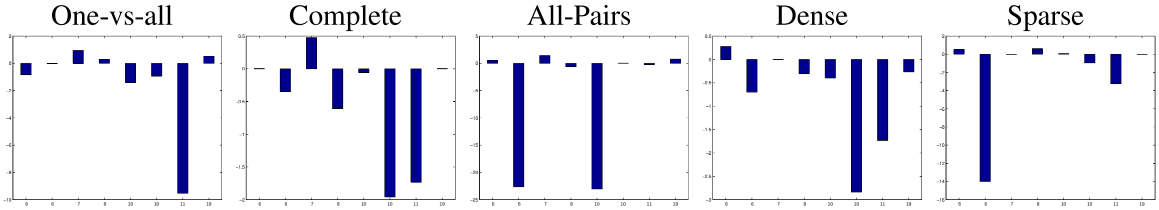

Figure 5: Comparison of the test error using Hamming decoding and loss-based decoding when the binary learner is support vector machines.

with randomized predictions. It is clear from the plots that loss-based decoding often gives better results than Hamming decoding for both SVM and AdaBoost. The difference is sometimes very significant. For instance, for the dataset Satimage, with the all-pairs code, SVM achieves 27.5% error with loss-based decoding while Hamming decoding results in an error rate of 50.4%. Similar results are obtained for AdaBoost. The difference is especially significant for the one-against-all and random dense codes. Note, however, that loss-based decoding based on AdaBoost with randomized predictions does not yield as good results as the straightforward use of loss-based decoding for AdaBoost with the exponential loss. This might be partially explained by the fact that AdaBoost with randomized predictions is not directly attempting to minimize the loss it uses for decoding.

To conclude the section, we discuss a set of experiments that compared the performance of the different output codes. In Figures 6 and 7, we plot the test error difference for pairs of codes using loss-based decoding with SVM and AdaBoost as the binary learners. Each plot consist of a 5×5 matrix of bar graphs. The rows and columns correspond, in order, to the five coding methods, namely, one-against-all, all-pairs, complete, random dense and random sparse. The bar graph in row i and column j shows the difference between the test error of coding method i minus the test error of coding method j for the datasets tested.

For SVM, it is clear that the widely used one-against-all code is inferior to all the other codes we tested. (Note that many of the bars in the top row of Figure 6 correspond to large positive values.) One-against-all often results in error rates that are much higher than the error rates of other codes. For instance, for the dataset Yeast, the one-against-all code has an error rate of 72% while the error rate of all the other codes is no more than 47.1% (random sparse) and can be as low as 39.6%

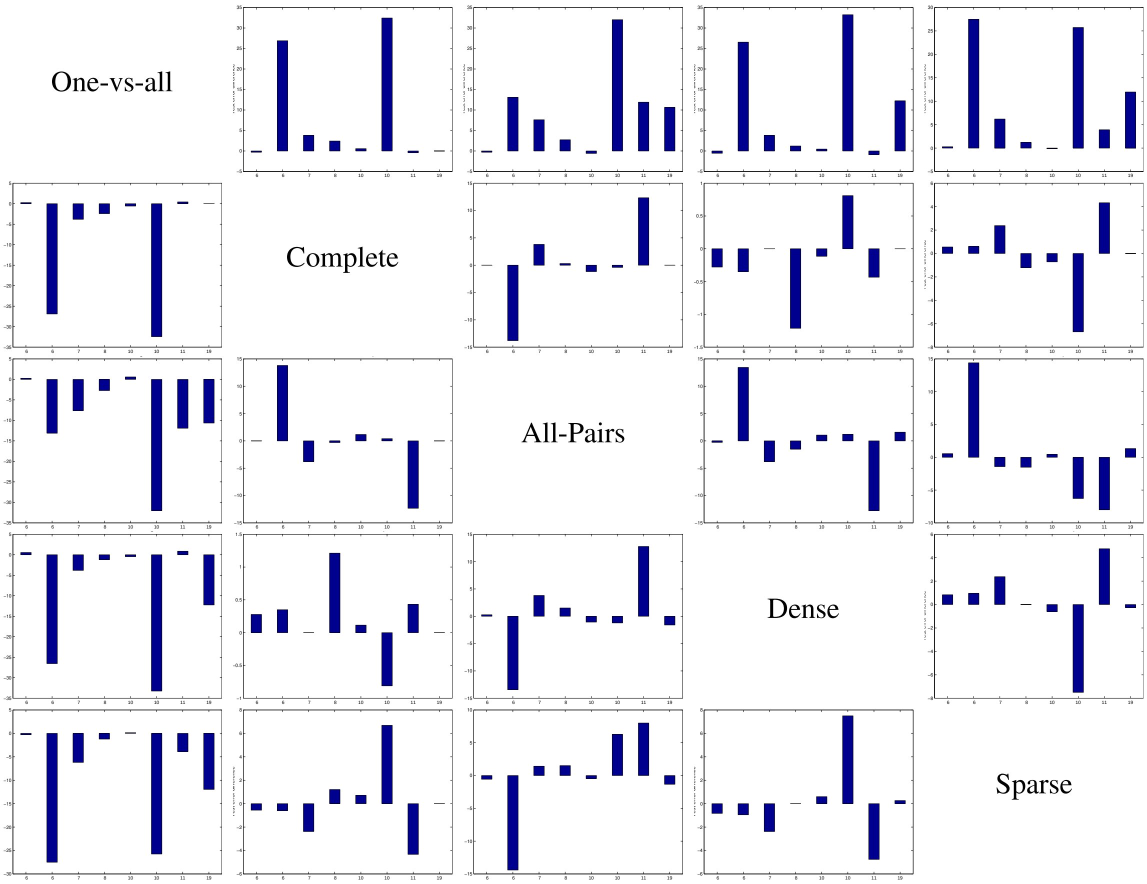

Figure 6: The difference between the test errors for pairs of error correcting matrices using support vector machines as the binary learners.

(random dense). On the very few cases that the one-against-all performs better than one of the other codes, the difference is not statistically significant. However, there is no clear winner among the four other output codes. For AdaBoost, none of the codes is persistently better than the other and it seems that the best code to use is problem dependent. These results suggest that an interesting direction for research would be methods for designing problem specific output codes. Some recent progress in this direction was made by Crammer and Singer (2000).

Acknowledgment

Most of the research on this paper was done while all three authors were at AT&T Labs. Thanks to the anonymous reviewers for their careful reading and helpful comments. Part of this research was supported by the Binational Science Foundation under grant number 1999038.

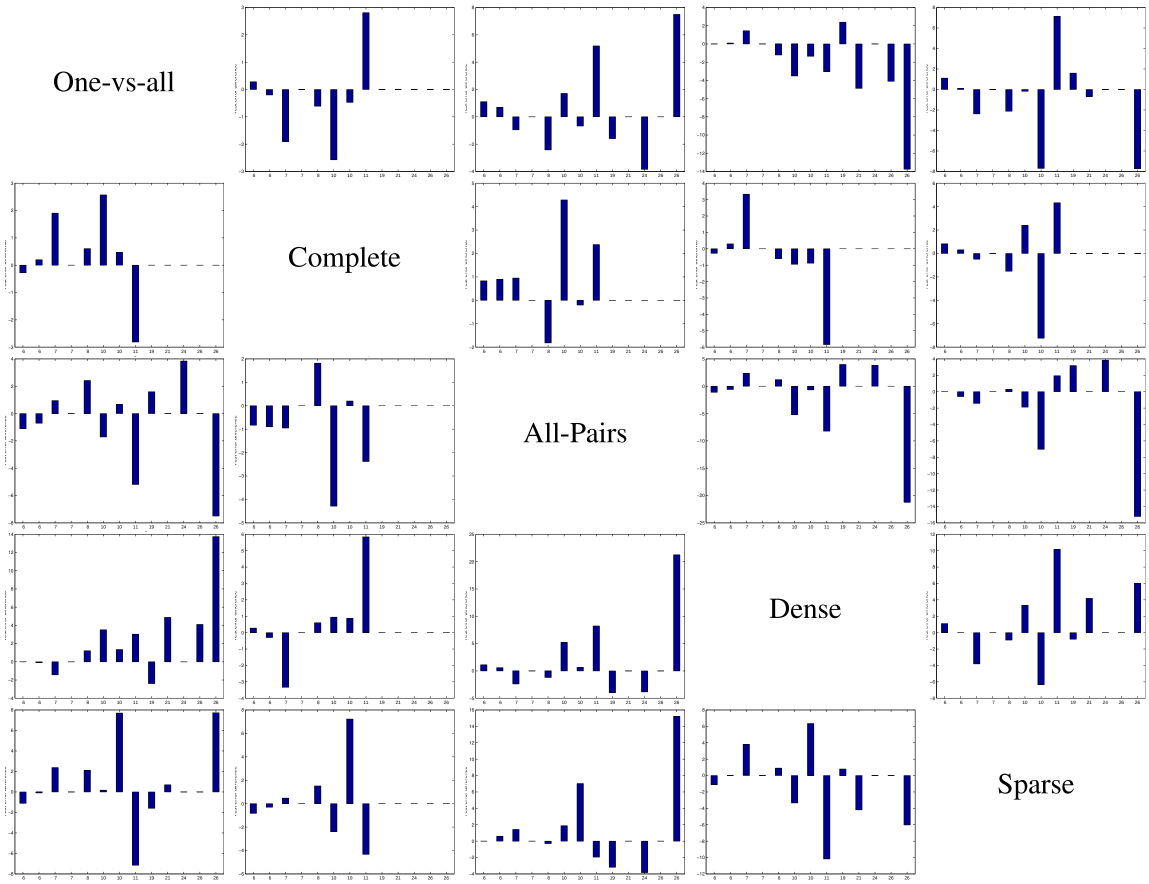

Figure 7: The difference between the test errors for pairs of error correcting matrices using AdaBoost for the binary learner.

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.

Breiman, L. (1997a). Arcing the edge. Tech. rep. 486, Statistics Department, University of California at Berkeley.

Breiman, L. (1997b). Prediction games and arcing classifiers. Tech. rep. 504, Statistics Department, University of California at Berkeley.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth & Brooks.

Collins, M., Schapire, R. E., & Singer, Y. (2000). Logistic regression, AdaBoost and Bregman distances. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Crammer, K., & Singer, Y. (2000). On the learnability and design of output codes for multiclass problems. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*.

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

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence*, *19*(4), 1–13.

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

Freund, Y. (1999). An adaptive version of the boost by majority algorithm. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pp. 102–113.

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.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, *38*(2), 337–374.

Guruswami, V., & Sahai, A. (1999). Multiclass learning, boosting, and error-correcting codes. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pp. 145–155.

Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. *The Annals of Statistics*, *26*(2), 451–471.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*(301), 13–30.

Höffgen, K.-U., & Simon, H.-U. (1992). Robust trainability of single neurons. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp. 428–439.

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*.

Lafferty, J. (1999). Additive models, boosting and inference for generalized divergences. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pp. 125–133.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). Functional gradient techniques for combining hypotheses. In Smola, A. J., Bartlett, P. J., Schölkopf, B., & Schuurmans, D. (Eds.), *Advances in Large Margin Classifiers*. MIT Press.

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

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rätsch, G., Onoda, T., & Müller, K.-R. (to appear). Soft margins for AdaBoost. *Machine Learning*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., & McClelland, J. L. (Eds.), *Parallel Distributed Processing – Explorations in the Microstructure of Cognition*, chap. 8, pp. 318–362. MIT Press.

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. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.

Schölkopf, B., Smola, A., Williamson, R., & Bartlett, P. (1998). New support vector algorithms. Tech. rep. NC2-TR-1998-053, NeuroColt2.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.