Journal of Experimental and Theoretical Artificial Intelligence
Vol. X, No. X, Month 200X, 000–000

Transforming examples for multiclass boosting

Tom Bylander*

Department of Computer Science, University of Texas at San Antonio, San Antonio, Texas, USA

Abstract

AdaBoost.M2 and AdaBoost.MH are boosting algorithms for learning from multiclass datasets. They have received less attention than other boosting algorithms because they require base classifiers that can handle the pseudoloss or Hamming loss, respectively. The difficulty with these loss functions is that each example is associated with k weights, where k is the number of classes. We address this issue by transforming an m-example dataset with k weights/example into a dataset with km examples and one weight/example. Minimizing error on the transformed dataset is equivalent to minimizing loss on the original dataset. Resampling the transformed dataset can be used for time efficiency and base classifiers that cannot handle weighted examples. We empirically apply the transformation on several multiclass datasets using naive Bayes and decision trees as base classifiers. Our experiment shows that it is competitive with AdaBoost.ECC, a boosting algorithm using output coding.

Keywords*:* Ensemble Methods, Boosting, Multiclass Learning, Pseudoloss, Hamming Loss

11. Introduction

AdaBoost.M2 (Freund and Schapire 1997) and AdaBoost.MH (Schapire and Singer 1999) are boosting algorithms for learning from multiclass datasets. Other multiclass boosting algorithms, such as AdaBoost.M1 (Freund and Schapire 1997) and boosting using output coding (Schapire 1997; Schapire and Singer 1999; Guruswami and Sahai 1999; Sun, Todorovic, Li and Wu 2005; Li 2006) have been well-studied within machine learning, while AdaBoost.M2 and AdaBoost.MH has seen very little study. The main reason for this contrast is the different requirements for base classifiers.

11.1. Requirements for AdaBoost.M1 and boosting with output coding

AdaBoost.M1 requires that the base classifier be able to input a multiclass dataset and a distribution on that dataset and to output a hypothesis that maps from instances to labels with an error rate less than 50% on the distribution. If a base classifier cannot input a distribution on the dataset, then resampling is an effective technique (Freund and Schapire 1996).

If a base classifier can only input a two-class dataset, output coding can be used to map a multiclass dataset into a two-class dataset. Classifying more than two classes can be handled by employing different codes over different boosting iterations (Schapire 1997; Guruswami and Sahai 1999; Sun et al. 2005; Li 2006) or within a single iteration (Schapire and Singer 1999). It turns out that boosting

* Email: bylander@cs.utsa.edu

with coding is better than AdaBoost.M1 even for base classifiers that can handle multiclass datasets. If a base hypothesis has an error rate of ½ or more, AdaBoost.M1 will stop, returning a final hypothesis based on the previous boosting iterations. However, a base hypothesis with a high error rate might still be informative relative to the distribution. Boosting with output coding, because the mapped datasets have only two classes, avoids this problem.

21.2. AdaBoost.M2 and AdaBoost.MH requirements

One difference in AdaBoost.M2's and AdaBoost.MH's requirements is that a base classifier must also be able to input a label weighting function that maps from examples and labels to weights. This results in k weights for each example. In AdaBoost.M2 and MH, the label weighting function is used to respectively compute the *pseudoloss* and the *Hamming loss*. For both loss functions, the loss for an incorrect prediction is based on the weight of the incorrect label, and, additionally for the Hamming loss, the weight of the correct label.

The other difference is that both AdaBoost.M2 and AdaBoost.MH can tolerate higher error rates from base hypotheses. For example, a pseudoloss less than ½ roughly corresponds to classification error less than $(k - 1)/k$.

Although it might not be difficult to revise a base classifier to handle the pseudoloss or Hamming loss, it would appear that the inconvenience and the fact that the base classifier must be changed at all has limited the use of AdaBoost.M2 and MH (Schapire 1997). For example, a revised base classifier makes it more difficult to compare different boosting and ensemble algorithms. In addition, the success of boosting with output coding appears to have lowered interest in AdaBoost.M2 and MH.

31.3. Summary of results

In this paper, we develop a technique to transform the examples so that multiclass base classifiers such as naive Bayes and C4.5 can be used for AdaBoost.M2 and MH. In the transformation, an m-example dataset with a distribution and a label weighting function is mapped into a dataset with km examples and one weight/example. Minimizing error on the transformed dataset is equivalent to minimizing pseudoloss (or Hamming loss) on the original dataset, so that the base classifier does not need to be modified for AdaBoost.M2 and MH.

Resampling the transformed dataset can be used for boosting base classifiers that cannot handle weighted examples (Freund and Schapire 1997). Resampling is also useful for achieving time efficiency comparable to AdaBoost.M1 and boosting with output codes.

In an empirical study, we applied the transformation to several multiclass datasets using naive Bayes (Duda and Hart 1973) and C4.5 (Quinlan 1993) as base classifiers. On the choice of using resampling, resampling with more (500) boosting iterations is more efficient and more accurate than no resampling with fewer (100) boosting iterations. We also show AdaBoost.M2 and MH with our transformation and resampling is competitive with AdaBoost.ECC (Guruswami and Sahai 1999; Sun et al. 2005).

First, we describe AdaBoost.M2 and the pseudoloss. Then, we demonstrate how to transform examples in AdaBoost.M2 for multiclass base classifiers. Next, we apply the same technique to AdaBoost.MH and the Hamming loss. Finally, we compare the empirical performance of AdaBoost.M2 and MH to AdaBoost.ECC.

22. AdaBoost.M2 and the pseudoloss

This section is based on the discussion and results from Freund and Schapire (1996; 1997).

A dataset S is a sequence of m examples $(x_1, y_1), ..., (x_m, y_m)$, where x_i is an instance from a domain X, and y_i is a label from a finite set Y, $|Y| = k \geq 2$. S is a multiclass dataset if $k > 2$, a binary dataset if $k = 2$.

12.1. Pseudoloss

AdaBoost.M2 uses a measure of loss called the *pseudoloss*. For a given hypothesis h and example (x_i, y_i), the pseudoloss is defined as follows:

$$\text{ploss}_q(h,i) = \tfrac{1}{2}\left(1 - h(x_i, y_i) + \sum_{y \neq y_i} q(i,y)\, h(x_i, y)\right) \tag{1}$$

$h : X \times Y \rightarrow [0,1]$ is a hypothesis. Unlike Freund and Schapire, we will require for our transformation that $1 = \Sigma_{y \in Y}\, h(x,y)$ for any instance x. In the case where h makes definite predictions, then if h predicts that y is the label for x, then $h(x, y) = 1$, else if h predicts some other label for x, then $h(x, y) = 0$. h making no prediction on x can be represented by $h(x, y) = 1/k$ for each $y \in Y$. More generally, h can provide a probabilistic prediction.

$Q : \{1, ..., m\} \times Y \rightarrow [0,1]$ is the *label weighting function*, such that for all i,

$$\sum_{y \neq y_i} q(i,y) = 1 \tag{2}$$

For each incorrect label y, $q(i, y)$ indicates the relative importance of differentiating between y and y_i.

The pseudoloss behaves as follows. If $h(x_i, y_i) = 1$, then there is no pseudoloss, and the values of $q(i, y)$ have no effect. However, if $h(x_i, y) = 1$ for an incorrect label y, the pseudo-loss will be $(1 + q(i, y))/2$, between ½ and 1 with a higher value if it is more important to avoid predicting y.

With a distribution $D(i) : \{1, ..., m\} \rightarrow [0,1]$ over the dataset S, the total pseudoloss will be:

$$\text{ploss}_{D,q}(h,S) = \sum_{i=1}^{m} D(i)\, \text{ploss}_q(h,i) \tag{3}$$

When, in the course of boosting, the examples have different values for D, the value of $D(i)$ is the weight for example i, and the values of $q(i, y)$ are the subweights for the incorrect labels on that example. As a convenience, one can construct a two argument distribution $D : \{1, ..., m\} \times Y \rightarrow [0,1]$ combining $D(i)$ and $q(i, y)$ so that:

$$D(i,y) = \begin{cases} D(i)\ q(i,y) & \text{if } y \neq y_i \\ 0 & \text{if } y = y_i \end{cases} \tag{4}$$

which implies:

$$\text{ploss}_{D,q}(h,S) = \sum_{i=1}^{m} \sum_{y \neq y_i} D(i,y)\ (1 - h(x_i, y_i) + h(x_i, y))/2 \tag{5}$$

22.2. AdaBoost.M2

AdaBoost.M2
Input:
a dataset $S = ((x_1, y_1), ..., (x_m, y_m))$ where $y_i \in Y$ and $|Y| = k \geq 2$
a base classifier **BaseLearn**
an integer T, the number of iterations
Initialize:
$D_1(i, y) = 1/(m(k-1))$ for $i \in \{1, ..., m\}$ and $y \in Y - \{y_i\}$
Do for $t = 1, ..., T$
1. Call **BaseLearn**, providing it with D_t
2. Get back a hypothesis $h_t : X \times Y \rightarrow [0,1]$.
3. Calculate the pseudoloss of h_t:

$$\varepsilon_t = \sum_{i=1}^{m} \sum_{y \neq y_i} D_t(i,y)\ (1 - h_t(x_i, y_i) + h_t(x_i, y))/2$$

4. Set $\beta_t = \varepsilon_t/(1 - \varepsilon_t)$
5. Update D_t (Z_t is a normalization constant)

$$D(i,y) = D(i,y)\ \beta_t^{(1+h_t(x_i,y_i)-h_t(x_i,y))/2} / Z_t$$

Output the final hypothesis

$$h_f(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \log(1/\beta_t)\ h_t(x,y)$$

Figure 1: The AdaBoost.M2 Algorithm

The AdaBoost.M2 algorithm (Figure 1) repeatedly calls a base classifier with the current distribution D_t, obtaining a sequence of hypotheses $h_1, ..., h_T$. After each call, the pseudoloss ε_t of h_t and the distribution update parameter β_t are calculated. The key to the algorithm is the distribution update in step 5 of the loop. This guarantees that the error of the final hypothesis h_f on the training examples has an upper bound of

$$(k-1)\prod_{t=1}^{T}(1-r^2)^{1/2} \tag{6}$$

where $r_t = 1 - 2\varepsilon_t$ (Freund and Schapire 1997).

33. Transforming examples for multiclass base classifiers

As previously discussed, base classifiers are not typically developed to handle the pseudoloss, instead they might handle one weight/example or assume each example is weighted equally. In this section, we show how to transform a pseudoloss distribution on the original dataset into a distribution on a modified dataset with one weight/example. As with the original boosting algorithms, resampling can be applied if the base classifier assumes each example is weighted equally (Freund and Schapire 1996). We assume that the base classifier can be applied to multiclass datasets. If this is not the case, then boosting with output coding would be a more suitable approach.

13.1. Derivation of transformation

Our idea is to transform each original example into k examples, one for each label, so that classification loss on the transformed examples will correspond to pseudoloss on the original examples. The weights on the transformed examples will be assigned to achieve this goal.

For an m-example dataset S and label weighting function $q(i, y)$, define an *example weighting function* $r : \{1, ..., m\} \times Y \rightarrow [0,1]$ as follows:

$$r(i,y) = \begin{cases} 2/k & \text{if } y = y_i \\ (1-q(i,y))/k & \text{if } y \neq y_i \end{cases} \tag{7}$$

$r(i, y)$ will be the weight for example (x_i, y) relative to the other possible labels. The division by k ensures that the $r(i, y)$ values for example i sum up to 1.

Lemma 1: $1 = \sum_{y \in Y} r(i,y)$

Proof:

By definition, $1 = \Sigma_{y \neq y_i} q(i,y)$ so it follows that:

$$\begin{aligned}\sum_{y \in Y} r(i,y) &= 2/k + \sum_{y \neq y_i} (1 - q(i,y))/k \\ &= (2 + (k-1) - \sum_{y \neq y_i} q(i,y))/k \\ &= (2 + (k-1) - 1)/k = k/k = 1\end{aligned}$$

■

For a given hypothesis h and example weighting function r define the loss on example i as:

$$\text{loss}_r(h,i) = \sum_{y \in Y} r(i,y)\ (1 - h(x_i, y)) \tag{8}$$

This is simply the classification loss. If $h(x_i, y) = 0$, then there will be a loss of $r(i, y)$. With this loss function, r satisfies the following property:

Lemma 2: For any hypothesis h, where $1 = \Sigma_{y \in Y} h(x,y)$ for any $x \in X$:

$$2\,\text{ploss}_q(h,i) = 2 - k + k\,\text{loss}_r(h,i)$$

Proof:

Starting with Equations (1) and (7):

$$\begin{aligned}2\,\text{ploss}_q(h,i) &= 1 - h(x_i, y_i) + \sum_{y \neq y_i} q(i,y)\ h(x_i, y) \\ &= 1 - h(x_i, y_i) + \sum_{y \neq y_i} (1 - k\ r(i,y))\ h(x_i, y) \\ &= 1 - h(x_i, y_i) + \sum_{y \neq y_i} h(x_i, y) - k \sum_{y \neq y_i} r(i,y)\ h(x_i, y)\end{aligned}$$

Noting that $1 = \Sigma_{y \in Y} h(x_i, y)$ implies $1 - h(x,y) = \Sigma_{y \neq y_i} h(x_i, y)$ we continue:

$$\begin{aligned}&= 2 - 2h(x_i, y_i) - k \sum_{y \neq y_i} r(i,y)\ h(x_i, y) \\ &= 2 - k\ r(x_i, y_i)\ h(x_i, y_i) - k \sum_{y \neq y_i} r(i,y)\ h(x_i, y) \\ &= 2 - k \sum_{y \in Y} r(i,y)\ h(x_i, y)\end{aligned}$$

Using Lemma 1 and Equation (8), we continue:

$$\begin{aligned} &= 2 - k + k \sum_{y \in Y} r(i,y) - k \sum_{y \in Y} r(i,y)\ h(x_i, y) \\ &= 2 - k + k \sum_{y \in Y} r(i,y)\ (1 - h(x_i, y)) \\ &= 2 - k + k \operatorname{loss}_r(h,i) \end{aligned}$$

■

Define $\operatorname{loss}_{D,r}(h, S)$ on all transformed examples as:

$$\operatorname{loss}_{D,r}(h,S) = \sum_{i=1}^{m} D(i) \operatorname{loss}_r(h,i) \tag{9}$$

Because $\operatorname{ploss}_q(h, i)$ linearly corresponds to $\operatorname{loss}_r(h,i)$, it follows that $\operatorname{ploss}_{D,q}(h,S)$ linearly corresponds to $\operatorname{loss}_{D,r}(h,S)$.

Theorem 3: For any hypothesis h, where $1 = \Sigma_{y \in Y}\ h(x, y)$ for any $x \in X$,

$$2 \operatorname{ploss}_{D,q}(h,S) = 2 - k + k \operatorname{loss}_{D,q}(h,S)$$

Proof:

$$\begin{aligned} 2 \operatorname{ploss}_{D,q}(h,S) &= \sum_{i=1}^{m} D(i)\, 2 \operatorname{ploss}_q(h,i) \\ &= \sum_{i=1}^{m} D(i)\ (2 - k + k \operatorname{loss}_r(h,i)) \\ &= 2 - k + k \sum_{i=1}^{m} D(i) \operatorname{loss}_r(h,i) \\ &= 2 - k + k \operatorname{loss}_{D,r}(h,S) \end{aligned}$$

■

It follows that pseudoloss values of 0, ½ and 1 respectively correspond to transformation loss values of $(k-2)/k$, $(k-1)/k$, and 1. Note a pseudoloss of ½ or less can be achieved by choosing the label with the highest prior according to D_r.

23.2. Application of transformation

For a distribution D and example weighting function r on an m-example dataset S, we define a distribution $D_r : \{1, ..., m\} \times Y \rightarrow [0,1]$:

$$D_r(i,y) = D(i)\ r(i,y) \tag{10}$$

$D_r(i, y)$ will be the weight for an example (x_i, y) where i $\in \{1, ...,m\}$ and $y \in Y$. As a result, any learning algorithm that minimizes loss on $D_r(i, y)$ will simultaneously minimize pseudoloss on distribution D with label weighting function q.

There are two disadvantages to this transformation. One is that m examples are transformed to km examples, possibly creating a much larger dataset along with duplication of instances. Another is that any hypothesis will have a $\text{loss}_{D, r}$ of at least $(k - 2)/k$, and any difference between hypotheses will be a factor of $k/2$ smaller than the corresponding difference in pseudoloss.

The large-dataset disadvantage can be addressed, to some extent, in a variety of ways. One can ensure that the transformed dataset only requires additional space for the km weights by not making deep copies of instances. Also, resampling can be applied to generate datasets with much fewer than km examples. Finally, an alternative is to modify the base classifier for minimizing pseudoloss along the lines of the transformation.

The higher loss is especially a disadvantage for resampling. When more weight is on incorrect labels, there will be more "noise" in the sample. That is, resampling will be more likely to draw incorrect labels and less likely to draw correct labels. This can be partially addressed with a modification r' of the example weighting function r:

$$r'(i, y) = r(i, y) - \min_{y' \in Y} r(i, y') \tag{11}$$

r' takes advantage of the property that the difference in the $r(i, \cdot)$ values is what leads to the linear relationship between $\text{ploss}_{D, q}$ and $\text{loss}_{D, r}$. A distribution similar to D_r can be easily defined from D and r', specifically:

$$D_r'(i, y) = D(i)\, r'(i, y) / Z \tag{12}$$

where Z is a normalization constant. When this is done, the first iteration of AdaBoost.M2 will have zero weights for incorrect labels, and following iterations will have reduced weights. Also, a pseudoloss of ½ or less can still be achieved by choosing the label with the highest prior according to D_r'.

44. Transformation for AdaBoost.MH and the Hamming loss

AdaBoost.MH is a general boosting algorithm for multilabel classification learning, i.e., where each example may have multiple labels (Schapire and Singer 1999). For example, a news article about DRM software might be labeled as about both technology and intellectual property. In this section, we focus on the specialized version of AdaBoost.MH for multiclass learning (Schapire and Singer 1999) with some changes in notation.

AdaBoost.MH
Input:
a dataset $S = ((x_1, y_1), ..., (x_m, y_m))$ where $y_i \in Y$ and $|Y| = k \geq 2$
a base classifier **BaseLearn**
an integer T, the number of iterations
Initialize:
$D_1(i, y_i) = 1/(2m)$ for $i \in \{1, ..., m\}$
$D_1(i, y) = 1/(2m(k-1))$ for $i \in \{1, ..., m\}$ and $y \neq y_i$
Do for $t = 1, ..., T$
1. Call **BaseLearn**, providing it with D_t
2. Get back a hypothesis $h_t : X \times Y \rightarrow [0,1]$.
3. Calculate the Hamming loss of h_t:

$$\varepsilon_t = \sum_{i=1}^{m} \sum_{y \in Y} D_t(i, y) E(h_t, i, y)$$

4. Set $\beta_t = (\varepsilon_t/(1-\varepsilon_t))^{1/2}$
5. Update D_t

$$D_{t+1}(i, y) = D_t(i, y)\, \beta_t^{-E(h_t, i, y)} / Z_t$$

where Z_t is a normalization constant.

Output the final hypothesis

$$h_f(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \log(1/\beta_t)\, h_t(x, y)$$

Figure 2: The AdaBoost.MH Algorithm

The AdaBoost.MH algorithm (Figure 2) repeatedly calls a base classifier with the current distribution D_t, obtaining a sequence of hypotheses $h_1, ..., h_T$. We require that $h(x, y) \geq 0$ and $1 = \Sigma_{y \in Y}\, h(x, y)$ for any h and x. After each call, the Hamming loss ε_t of h_t and the distribution update parameter β_t are calculated. The notation $E(h_t, i, y)$ denotes the error of h_t on example i with respect to label y:

$$E(h_t, i, y) = \begin{cases} 1 - h_t(x_i, y) & \text{if } y = y_i \\ h_t(x_i, y) & \text{if } y \neq y_i \end{cases} \qquad (13)$$

The Hamming loss sums $D_t(i, y)$ where h_t was wrong for label y, i.e., if $h_t(x_i, y) = 1$ and $y \neq y_i$, then $D(i, y)$ and $D(i, y_i)$ are included in the loss. The distribution update in step 5 of the loop guarantees that the error of the final hypothesis h_f on the training examples has an upper bound of

$$(k-1)^{1/2}\prod_{t=1}^{T}(1-r_t^2)^{1/2} \tag{14}$$

where $r_t = 1 - 2\varepsilon_t$ (Schapire and Singer 1999).

The idea of the transformation for AdaBoost.MH (to transform each original example into k examples) is similar to the transformation for AdaBoost.M2. As before, the weights on the transformed examples will ensure that the classification loss on the transformed examples corresponds to the Hamming loss on the original examples.

Let $D(i, y)$ be a distribution on the dataset and labels. For convenience, let $D(i) = \Sigma_{y \in Y} D(i, y)$. Let $F(i)$ be any function such that $F(i) \geq D(i, y)$ for all i and $y \neq y_i$. For example, we could assign $F(i) = D(i)$. Then, we can define a distribution D_F for km examples (x_i, y) as follows:

$$D_F(i,y) = \begin{cases} (F(i) + D(i,y_i))/Z_{D,F} & \text{if } y = y_i \\ (F(i) - D(i,y_i))/Z_{D,F} & \text{if } y \neq y_i \end{cases} \tag{15}$$

$Z_{D,F}$ is a normalization factor so that $1 = \Sigma_{i=1}^{m} \Sigma_{y \in Y} D_F(i, y)$. To ensure that the weights on incorrect labels are as small as possible, we can choose $F(i) = \max_{y \neq y_i} D(i, y)$.

With this definition, the loss on D_F ($\text{loss}_{D,F}$) is linearly related to the Hamming loss on D (hloss_D).

Theorem 4: For any hypothesis h, where $1 = \Sigma_{y \in Y} h(x, y)$ for any $x \in X$, and D_F and $Z_{D,F}$ as defined above:

$$Z_{D,F}\,\text{loss}_{D,F}(h,S) = \text{hloss}_D(h,S) - 1 + \sum_{i=1}^{m}((k-1)F(i) + D(i,y_i))$$

Proof:

$$\begin{aligned} Z_{D,F}\,\text{loss}_{D,F}(h,S) &= Z_{D,F}\sum_{i=1}^{m}\sum_{y \in Y}(1-h(x_i,y))\,D_F(i,y) \\ &= \sum_{i=1}^{m}(1-h(x_i,y_i))(F(i)+D(i,y_i)) \\ &\quad + \sum_{i=1}^{m}\sum_{y \neq y_i}(1-h(x_i,y))(F(i)-D(i,y)) \end{aligned}$$

$$
\begin{aligned}
= & \sum_{i=1}^{m} \sum_{y \in Y} (1 - h(x_i, y))\, F(i) \\
& + \sum_{i=1}^{m} (1 - h(x_i, y_i))\, D(i, y_i) \\
& - \sum_{i=1}^{m} \sum_{y \neq y_i} (1 - h(x_i, y))\, D(i, y) \\
= & \sum_{i=1}^{m} F(i) \sum_{y \in Y} (1 - h(x_i, y)) + \sum_{i=1}^{m} (1 - h(x_i, y_i))\, D(i, y_i) \\
& + \sum_{i=1}^{m} \sum_{y \neq y_i} h(x_i, y)\, D(i, y) - \sum_{i=1}^{m} \sum_{y \neq y_i} D(i, y)
\end{aligned}
$$

The second and third summations together equal the Hamming loss. Also, $1 = \Sigma_{y \in Y}\, h(x, y)$ implies $k - 1 = \Sigma_{y \in Y}\, (1 - h(x, y))$. Finally $\Sigma_{y \neq y_i}\, D(i, y) = D(i) - D(i, y_i)$. As a result, we continue:

$$
\begin{aligned}
= & \sum_{i=1}^{m} F(i)\, (k - 1) + \mathrm{hloss}_D(h, S) - \sum_{i=1}^{m} (D(i) - D(i, y_i)) \\
= & \ \mathrm{hloss}_D(h, S) - 1 + \sum_{i=1}^{m} ((k - 1)\, F(i) + D(i, y_i))
\end{aligned}
$$

■

Unlike the transformation for AdaBoost.M2, the transformation for AdaBoost.MH does not result in a nice correspondence between Hamming loss and the transformation loss. For example, a transformation loss of $(k - 1)/k$ corresponds to a Hamming loss of ½ only if $\Sigma_{i=1}^{m}\, D(i, y) = ½$. This will only hold true for the first iteration of AdaBoost.MH; after that, the sum will increase. As boosting proceeds, progressively lower error rates are needed to achieve a Hamming loss less than ½.

55. Experiments

We applied AdaBoost.ECC (Guruswami and Sahai 1999; Sun et al. 2005), AdaBoost.M2T (AdaBoost.M2 with our transformation and resampling), and AdaBoost.MHT (transformation and resampling) to the 21 UCI datasets (Asuncion and Newman 2007) summarized in Table 1. We used naive Bayes and C4.5 as base classifiers. The class with the highest probability was used as the prediction. 500 boosting iterations were performed. We also performed 100 boosting iterations using AdaBoost.M2 with our transformation, but without resampling, to determine the effectiveness of resampling. Our evaluation was performed using stratified 10-fold cross-validation.

Table *1*: Datasets Used in Experiments

Dataset	Examples	Attributes	Classes
anneal	150	4	3
arrhythmia	452	280	16
audiology	226	69	24
autos	205	26	7
balance-scale	625	4	3
ecoli	336	7	8
glass	214	9	7
hypothyroid	3772	29	4
iris	150	4	3
letter	20000	16	26
lymph	148	18	4
optdigits	5620	64	10
pendigits	10992	16	10
primary-tumor	339	17	22
segment	2310	19	7
soybean	683	35	19
splice	3190	61	3
vehicle	846	18	4
vowel	990	13	11
waveform	5000	40	3
zoo	101	17	11

We applied AdaBoost.ECC (Guruswami and Sahai 1999; Sun et al. 2005), AdaBoost.M2T (AdaBoost.M2 with our transformation and resampling), and AdaBoost.MHT (transformation and resampling) to the 21 UCI datasets (Asuncion and Newman 2007) summarized in Table 1. We used naive Bayes and C4.5 as base classifiers. The class with the highest probability was used as the prediction. 500 boosting iterations were performed. We also performed 100 boosting iterations using AdaBoost.M2 with our transformation, but without resampling, to determine the effectiveness of resampling. Our evaluation was performed using stratified 10-fold cross-validation.

For M2T and MHT, boosting was halted if the base hypothesis was perfect or nearly so (pseudoloss for M2T or Hamming loss for MHT less than 10^{-6}). Also, up to 10 resamplings were performed to find a base hypothesis with pseudoloss (Hamming loss) less than ½. Boosting was halted if no such base hypothesis could be found.

The naive Bayes (Duda and Hart 1973) base classifier used Laplace's rule of succession and partitioned numeric attributes into 10 bins. The C4.5 (Quinlan 1993) base classifier was C4.5, release 8, with parameters at default settings. C4.5 was modified to input weights for examples. For an *m*-example dataset, the sum of the examples' weights was normalized to *m* for the naive Bayes and C4.5 base classifiers (counting examples corresponds to adding weights).

15.1. Resampling

While resampling is more efficient for M2T and MHT, it might be more effective to apply base classifiers to the full transformed set of *km* examples. For AdaBoost.M2, it turns out that resampling is just as effective with the same number of iterations with some improvement with more iterations.

Comparing 100 boosting iterations with and without resampling, resampling had a lower error rate on 16 of the 21 datasets using naive Bayes and 10 of the 21 datasets using C4.5 (data not shown). Comparing 100 boosting iterations without resampling to 500 with resampling, resampling had a lower error rate on 19 of the 21 datasets using naive Bayes and 11 of the 21 datasets using C4.5. In addition, 100 boosting iterations without resampling took over 80% and 160% more time compared to 500 iterations with resampling using naive Bayes and C4.5, respectively.

25.2. Comparison to AdaBoost.ECC

Table 2 displays the error rates of AdaBoost.ECC, M2T, and MHT on the datasets using naive Bayes and C4.5 as base classifiers. 500 boosting iterations were performed using each algorithm, although we note MHT halted before 500 iterations about half the time using naive Bayes and 3 datasets using C4.5.

AdaBoost.M2T and MHT were competitive with ECC with both base classifiers. With the naive Bayes base classifier, ECC had the lowest error rate on 9 datasets (including ties), M2T on 10 datasets, and MHT on 6 datasets. With the C4.5 base classifier, ECC had the lowest error rate on 12 datasets, M2T on 6 datasets, and MHT on 5 datasets. While ECC performed better using C4.5, either M2T or MHT did obtain the lowest error rate on almost half of the datasets. Exceptional performances include M2T using naive Bayes on the balance-scale dataset and MHT using naive Bayes on the primary-tumor dataset.

Table *2*: Error Rates for AdaBoost.MCC, .M2T and .MHT using Naive Bayes and C4.5

Dataset	**Naive Bayes**			C4.5		
	ECC	**M2T**	**MHT**	**ECC**	**M2T**	**MHT**
anneal	**0.67**	0.78	**0.67**	**2.67**	3.01	2.90
arrhythmia	30.53	**29.65**	29.87	24.34	27.21	**23.89**
audiology	**18.14**	22.57	20.35	15.93	15.93	**15.04**
autos	**18.05**	**18.05**	20.00	16.10	15.61	**14.63**
balance-scale	10.08	**7.52**	11.04	23.20	**22.88**	26.24
ecoli	20.24	**17.56**	19.35	14.29	**13.99**	14.88
glass	34.11	**25.70**	29.44	20.56	21.03	**20.09**
hypothyroid	**0.74**	**0.74**	1.03	**0.29**	0.27	**0.29**
iris	7.33	8.00	**6.67**	5.33	**4.67**	5.33
letter	25.38	**24.52**	28.50	**2.49**	3.46	2.69
lymph	14.86	**14.19**	16.89	**11.49**	14.19	15.54
optdigits	4.18	**4.11**	6.46	**1.17**	1.53	1.33
pendigits	**3.34**	4.22	7.31	**0.43**	0.64	0.51
primary-tumor	58.70	54.57	**53.39**	**57.23**	**57.23**	58.11
segment	**3.20**	3.64	6.28	**1.08**	1.21	1.47
soybean	7.03	7.61	**6.21**	**6.30**	6.59	6.44
splice	7.15	8.34	**6.89**	**4.58**	4.83	4.64
vehicle	28.72	**26.36**	32.51	**20.92**	21.28	21.63
vowel	**18.89**	21.01	20.51	**1.72**	4.44	2.93
waveform	19.28	18.54	**18.12**	**13.90**	15.00	14.44
zoo	**2.97**	4.95	3.96	3.96	**2.97**	**2.97**

66. Conclusions

Our transformation provides an effective technique for applying AdaBoost.M2 and AdaBoost.MH with any multiclass base classifier. The issue of base classifiers understanding pseudoloss or Hamming loss is addressed by converting a *m*-example *k*-class dataset with *k* weights/example to a dataset with *km* examples and one weight/example. We have shown that AdaBoost.M2T and MHT are competitive with AdaBoost.MCC using naive Bayes and C4.5 as base classifiers, though MCC does somewhat better on more datasets when C4.5 is the base classifier.

If resampling is not performed, the creation of a larger dataset leads to significantly longer learning times. For example, on the letter dataset (20000 examples, 26 classes) and using the same number of iterations, AdaBoost.M2T without resampling took more time than AdaBoost.M2T with resampling by a factor of about 20. Fortunately, AdaBoost.M2T with resampling using the same or more iterations was more efficient and just as effective than AdaBoost.M2T without resampling. All of this makes AdaBoost.M2T and MHT with resampling a good choice for applying boosting on multiclass datasets with multiclass base classifiers.

In the future, we expect that many of the same refinements that have been applied to boosting with output coding can also be advantageously applied to AdaBoost.M2T and MHT. In addition, we will experiment more with AdaBoost.M2T and MHT for their effectiveness on overfitting. In a sense, because weights are assigned to incorrect labels, AdaBoost.M2T and MHT introduce noise to the dataset; however, their performance was still comparable to AdaBoost.MCC. Another interest of ours is using rule learning as the base classifier. With M2T and MHT, it should be possible to construct rules in sequence, each rule for a specific class rather than an output code.

7References

Asuncion, A. and Newman, D.J. (2007), "UCI Machine Learning Repository," Department of Information and Computer Science, University of California at Irvine, *http://www.ics.uci.edu/~mlearn/MLRepository.html.*

Duda, R.O., and Hart, P.E. (1973), *Pattern Classification and Scene Analysi*s, New York: John Wiley.

Freund, Y., and Schapire, R.E. (1996), "Experiments with a New Boosting Algorithm," in *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 148-156.

Freund, Y., and Schapire, R.E. (1997), "A Decision-theoretic Generalization of On-line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, 55, 119-139.

Guruswami, V., and Sahai, A. (1999), "Multiclass Learning, Boosting, and Error-correcting Codes," in *Proceedings of the Twelfth Annual Conference on Computational Learning Theor*y, pp. 145-155.

Li, L. (2006), "Multiclass Boosting with Repartitioning," *Proceedings of the Twenty-Third International Conference on Machine Learning*, pp. 569-576.

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

Schapire, R.E. (1997), "Using Output Codes to Boost Multiclass Learning Problems," *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 313-321.

Schapire, R.E., and Singer, Y. (1999), "Improved Boosting Using Confidence-rated Predictions," *Machine Learning*, 37, 297-336.

Sun, Y., Todorovic, S., Li, J., and Wu, D. (2005), "Unifying the Error-correcting and Output-code AdaBoost within the Margin Framework," *Proceedings of the Twenty-Second International Conference on Machine Learning*, pp. 872-879.