Multiclass Boosting with Hinge Loss based on Output Coding

Tianshi Gao TIANSHIG@STANFORD.EDU
Electrical Engineering Department, Stanford, CA 94305 USA

Daphne Koller KOLLER@CS.STANFORD.EDU
Computer Science Department, Stanford, CA 94305 USA

Abstract

Multiclass classification is an important and fundamental problem in machine learning. A popular family of multiclass classification methods belongs to reducing multiclass to binary based on output coding. Several multiclass boosting algorithms have been proposed to learn the coding matrix and the associated binary classifiers in a problem-dependent way. These algorithms can be unified under a sum-of-exponential loss function defined in the domain of margins (Sun et al., 2005). Instead, multiclass SVM uses another type of loss function based on hinge loss. In this paper, we present a new output-coding-based multiclass boosting algorithm using the multiclass hinge loss, which we call HingeBoost.OC. HingeBoost.OC is tested on various real world datasets and shows better performance than the existing multiclass boosting algorithm AdaBoost.ERP, one-vs-one, one-vs-all, ECOC and multiclass SVM in a majority of different cases.

1. Introduction

Multiclass classification is an important and fundamental problem in machine learning. Due to the richness of concepts in many domains, a classifier is often required to discriminate far more than two categories. For example, in visual recognition, the SUN dataset (Xiao et al., 2010) contains 899 scene categories and ImageNet (Deng et al., 2009) has 15,589 synsets. Many of the research efforts in machine learning have been devoted to the binary classification problem, so it is very useful and important to investigate how to use well studied binary classification algorithms to solve multiclass problems.

Output coding is a general framework to reduce a multiclass problem into binary (Dietterich & Bakiri, 1995; Allwein et al., 2001). It unifies popular methods like one-vs-all and one-vs-one by viewing different choices of sets of binary classifiers as induced by a coding matrix. Given a K-class problem, the coding matrix has K rows each of which corresponds a codeword assigned to a class. A binary classifier is learned for each and every column of the matrix, on instances that are relabeled by the column. Different fixed coding matrices are compared in (Allwein et al., 2001) on various datasets, but none is the best on all datasets. The results in (Allwein et al., 2001) suggest that learning a coding matrix with the associated binary classifiers in a problem-dependent way is better than using a pre-defined coding matrix.

However, as shown by Crammer & Singer (2000), simultaneously finding a whole coding matrix with a set of binary classifiers is intractable. Alternatively, multiclass boosting algorithms based on output coding, i.e., AdaBoost.OC and AdaBoost.ECC, have been proposed by Schapire (1997) and Guruswami & Sahai (1999), where each column and its binary classifier are viewed as weak learners and are learned one at a time in a stage-wise way. Sun et al. (2005) unified these two methods by showing that both are actually performing stage-wise functional gradient descent on a loss function defined in the domain of margins. To trade-off the error-correcting property of the coding matrix and the performance of the binary learner, Li (2006) proposed to modify the coding matrix according to the learning ability of the binary learner, resulting in an improved boosting algorithm AdaBoost.ERP.

Most of previous output-coding-based multiclass boosting algorithms are unified under a loss based on the sum-of-exponential of the margins (Sun et al., 2005). Another type of multiclass classification loss used by multiclass SVM (Crammer & Singer, 2002) puts a truncated linear penalty on the margins, which

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

we refer to as multiclass hinge loss. There are several other methods that are based on the multiclass hinge loss and can be thought of as a relaxation of the discrete coding matrix to real values. Both Rätsch et al. (2003) and Amit et al. (2007) learn the real-valued coding matrix and the embedding functions jointly based on the hinge loss. The former used the alternating method to optimize the bi-convex problem and the latter proposed a convex reformulation. More generally, Zou et al. (2008) outlined a class of smooth convex loss functions that are Fisher-consistent, including logit loss, squared hinge loss, etc.

In this work, we propose a new output-coding-based multiclass boosting algorithm using the multiclass hinge loss, which we call HingeBoost.OC. HingeBoost.OC offers several advantages over previous methods. First, unlike previous output-coding-based methods, we use the multiclass hinge loss which is more robust than the traditional sum-of-exponential loss when there is noise/outlier (Hastie et al., 2009). Second, our framework naturally allows kernelization (some methods like (Zou et al., 2008) cannot do so), which can usually boost the performance significantly. Third, our framework can admit any user-defined cost functions, while many previous methods are not cost-sensitive. Finally, instead of requiring the weak learner to predict a single label from the entire label space (Zou et al., 2008), our framework focuses on reducing multiclass to binary, so we can take advantage of existing highly optimized binary models/softwares.

2. Analysis of Multiclass Loss Functions

In this section, we analyze and compare multiclass loss functions. For binary classification, many algorithms are formulated to minimize some loss functions over margins. For example, boosting uses an exponential loss over the margin and SVM adopts the hinge loss. The concept of margins can be naturally generalized to the multiclass setting. Given a function $f(\mathbf{x}, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, with the decision made by $\hat{y} = \text{argmax}_y f(\mathbf{x}, y)$, the *margin* of an instance $\mathbf{x}_i$ with respect to a label y is defined as (Sun et al., 2005; Li, 2006)

$$\rho_f(\mathbf{x}_i, y) = f(\mathbf{x}_i, y_i) - f(\mathbf{x}_i, y) \tag{1}$$

where y_i is the true label of $\mathbf{x}_i$. Note that $\rho_f(\mathbf{x}_i, y_i) = 0$. The existing multiclass boosting algorithms based on output coding (Schapire, 1997; Guruswami & Sahai, 1999; Sun et al., 2005; Li, 2006) are understood as a stage-wise optimization procedure to minimize a loss defined over the multiclass margins. This common loss is defined as

$$L_{\text{exp}}(f) = \frac{1}{N(K-1)} \sum_{i=1}^{N} \sum_{y:y \neq y_i} e^{-\rho_f(\mathbf{x}_i, y)} \tag{2}$$

where K is the number of classes. Another type of multiclass classification loss used by the multiclass SVM (Crammer & Singer, 2002) is given as

$$L_{\text{hinge}}(f) = \frac{1}{N} \sum_{i=1}^{N} \max_y \{1 - \rho_f(\mathbf{x}_i, y) - \delta_{y_i,y}\} \tag{3}$$

where $\delta_{y_i,y} = 1$, if $y_i = y$, otherwise 0. Note that $L_{\text{hinge}}(f) \geq 0$, since $1 - \rho_f(\mathbf{x}_i, y_i) - \delta_{y_i,y_i} = 0$. Furthermore, the classification error is denoted as

$$L(f) = \frac{1}{N} \sum_{i=1}^{N} (1 - \delta_{y_i,\hat{y}_i}) \tag{4}$$

where $\hat{y}_i = \text{argmax}_y f(\mathbf{x}_i, y)$.

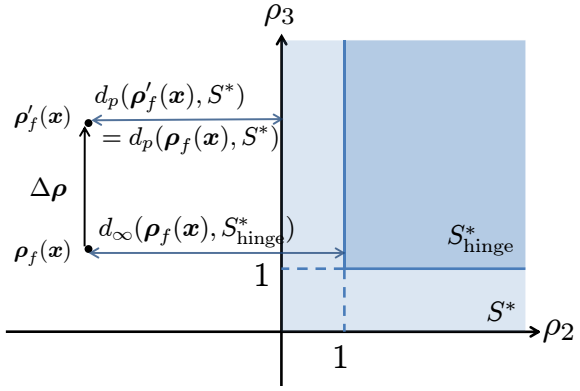

Figure 1. Illustration of the margin domain for a 3-class problem. We assume the true label of $\mathbf{x}$ is 1, so ρ_1 is always 0, which we don't show. Please refer to the text for more details.

We present a simple geometric interpretation of these losses on the domain of margins. Consider a dataset with one instance $\mathbf{x}$, and without loss of generality we assume its true label is 1. Then given some f, the *margin vector* is $\boldsymbol{\rho}_f(\mathbf{x}) = (0, \rho_f(\mathbf{x}, 2), \ldots, \rho_f(\mathbf{x}, K))^T$. Define $S^* = \{\boldsymbol{\rho} | \rho_1 = 0, \rho_k > 0, k = 2, \ldots, K, \boldsymbol{\rho} \in \mathbb{R}^K\}$. Obviously, if $\boldsymbol{\rho}_f(\mathbf{x}) \in S^*$, then $L(f) = 0$, otherwise $L(f) = 1$. Therefore, S^* is the optimal set on the margin domain which induces zero mis-classification error. Intuitively, to minimize $L(f)$, we want the learning algorithm to search the space of f to minimize the distance between the margin vector $\boldsymbol{\rho}_f(\mathbf{x})$ and the optimal set S^*, i.e., $\min_f d_p(\boldsymbol{\rho}_f(\mathbf{x}), S^*)$, where $d_p(\boldsymbol{\rho}_f(\mathbf{x}), S^*) = \min_{\boldsymbol{\rho} \in S^*} \| \boldsymbol{\rho}_f(\mathbf{x}) - \boldsymbol{\rho} \|_p$ for some chosen norm ℓ_p. Figure 1 shows the margin domain for $K = 3$ (ignoring the dimension of ρ_1, since it's always 0). The light blue region corresponds to S^*.

For the hinge loss, let $S^*_{\text{hinge}} = \{\boldsymbol{\rho} | \rho_1 = 0, \rho_k - 1 \geq 0, k = 2, \ldots, K, \boldsymbol{\rho} \in \mathbb{R}^K\}$, then $L_{\text{hinge}}(f) = 0$ if and only if $\boldsymbol{\rho}_f(\mathbf{x}) \in S^*_{\text{hinge}}$. The dark blue region in Figure 1 corresponds to S^*_{hinge}. One can show that $L_{\text{hinge}}(f) = d_\infty(\boldsymbol{\rho}_f(\mathbf{x}), S^*_{\text{hinge}})$, where

$d_\infty(\boldsymbol{\rho}_f(\mathbf{x}), S^*_{\text{hinge}}) = \min_{\boldsymbol{\rho}\in S^*_{\text{hinge}}} \parallel \boldsymbol{\rho}_f(\mathbf{x}) - \boldsymbol{\rho} \parallel_\infty$. Therefore, minimizing the hinge loss is equivalent to minimizing the distance (with infinity norm) between the margin vector $\boldsymbol{\rho}_f(\mathbf{x})$ and the optimal set S^*_{hinge}. Furthermore, if $\boldsymbol{\rho}_f(\mathbf{x}) \notin S^*$, then it's easy to show that $d_\infty(\boldsymbol{\rho}_f(\mathbf{x}), S^*) = d_\infty(\boldsymbol{\rho}_f(\mathbf{x}), S^*_{\text{hinge}}) - 1$. This implies that the margin vector gets closer to the optimal set S^* of mis-classification error if we decrease L_{hinge}.

However, decreasing $L_{\exp}(f)$ does not necessarily make $\boldsymbol{\rho}_f(\mathbf{x})$ closer to S^*. Consider the point $\boldsymbol{\rho}_f(\mathbf{x})$ in Figure 1. If we take a step Δf such that $\Delta\boldsymbol{\rho} = (0, 0, \alpha)$, where $\alpha > 0$, then $L_{\exp}(f + \Delta f) < L_{\exp}(f)$ (since $\rho'_f(\mathbf{x}, 3) > \rho_f(\mathbf{x}, 3) \Rightarrow e^{-\rho'_f(\mathbf{x},3)} < e^{-\rho_f(\mathbf{x},3)}$, where $\boldsymbol{\rho}'_f(\mathbf{x}) = \boldsymbol{\rho}_f(\mathbf{x}) + \Delta\boldsymbol{\rho}$). However, $d_p(\boldsymbol{\rho}_f, S^*) = d_p(\boldsymbol{\rho}'_f, S^*)$ for ℓ_p with different choices of p.

Another important difference is how sensitive the loss is to the noise/outlier. For $L_{\exp}$, it puts exponential penalty on the violated margins, while L_{hinge} uses a linear penalty. This implies that if a high level of noise/outlier presents, L_{hinge} should be more robust than $L_{\exp}$ (Hastie et al., 2009).

In some domains, the 0-1 type loss (4) does not capture some intrinsic distance measures between classes. For example, in visual recognition, classifying a cat into a dog should be more tolerable than classifying a cat into a car. Therefore, we introduce a cost matrix $\mathcal{C}(y, y')$ specifying the cost of classifying an instance from class y into y'. In fact, (3) is a special case where $\mathcal{C}(y, y') = 1 - \delta_{y,y'}$. An example of a multiclass cost matrix can be found in (Deng et al., 2009), where a cost matrix is defined with the help of a visual hierarchy. Given a cost matrix $\mathcal{C}$, the empirical mis-classification cost becomes

$$L^{\mathcal{C}}(f) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{C}(y_i, \hat{y}_i) \tag{5}$$

Furthermore, we define a *cost-sensitive hinge loss* as

$$L^{\mathcal{C}}_{\text{hinge}}(f) = \frac{1}{N}\sum_{i=1}^{N} \max_y\{\mathcal{C}(y_i, y) - \rho_f(\mathbf{x}_i, y)\} \tag{6}$$

It can be shown that $L^{\mathcal{C}}_{\text{hinge}}$ is an upper bound on $L^{\mathcal{C}}$.

3. Multiclass Boosting based on Output Coding

Given the loss defined in the previous section, the goal of learning is to search in a function space to minimize the loss. As shown by Sun et al. (2005), the output-coding-based multiclass boosting algorithms are actually performing stage-wise functional gradient descent to minimize (2). In the following, we briefly introduce the general framework of multiclass boosting based on output coding. Given a set of training instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $y_i \in \mathcal{Y} = \{1, 2, \ldots, K\}$, the goal is to learn a discriminative function $f(\mathbf{x}, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ with the decision made as $\hat{y} = \text{argmax}_y f(\mathbf{x}, y)$. In the boosting framework, f is a weighted average of multiple weak learners, i.e., $f(\mathbf{x}, y) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x}, y)$, where $h_t(\mathbf{x}, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. In the case of output-coding-based multiclass boosting (Schapire, 1997; Guruswami & Sahai, 1999; Sun et al., 2005; Li, 2006), the form of the weak learner is $h_t(\mathbf{x}, y) = \mu_t(y)g_t(\mathbf{x})$, where $\mu_t : \mathcal{Y} \to \{-1, 0, +1\}$ is a coloring function inducing a set of negative classes $S^- = \{(\mathbf{x}_i, y_i)|\mu_t(y_i) = -1\}$ and a set of positive classes $S^+ = \{(\mathbf{x}_i, y_i)|\mu_t(y_i) = +1\}$, and $g_t(\mathbf{x})$ is a binary classifier trained to separate S^- and S^+. Usually, $M = [\mu_1, \mu_2, \ldots, \mu_T] \in \{-1, 0, +1\}^{K\times T}$ is called the codebook or coding matrix. The boosting algorithms usually start with an empty codebook M, and use a stage-wise way to learn a single column μ_t and its binary classifier $g_t(\mathbf{x})$ at a time.

4. Multiclass Boosting with Hinge Loss

Under the general framework of output-coding-based multiclass boosting, we introduce our new boosting algorithm based on multiclass hinge loss (HingeBoost.OC). We assume the binary classifier is $g_t(\mathbf{x}) = \mathbf{w}_t^T\phi(\mathbf{x}) + b_t$, where $\phi(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^D$ is a feature mapping given explicitly or implicitly (as shown later). Since our weak classifier outputs a real value, the role of α_t is absorbed into the parameters $\mathbf{w}_t$ and b_t , therefore, our final discriminative function is given as $f(\mathbf{x}, y) = \sum_{t=1}^T h_t(\mathbf{x}, y) = \sum_{t=1}^T \mu_t(y)g_t(\mathbf{x})$, where $\mu_t : \mathcal{Y} \to \{-1, +1\}$. We will first define a regularized objective based on the multiclass hinge loss, and optimize it in a stage-wise way, i.e., one $h_t(\mathbf{x}, y)$ at a time. To optimize over $h_t(\mathbf{x}, y) = \mu_t(y)g_t(\mathbf{x})$ at each round, we iteratively optimize over $\mu_t(y)$ or $g_t(\mathbf{x})$ with the other fixed. As we will show, optimizing over $g_t(\mathbf{x})$ is a margin-rescaled binary SVM problem with the number of constraints independent of the number of classes. To optimize over $\mu_t(y)$ we use a greedy descent method. Both steps can be solved efficiently.

4.1. Objective

Our goal is to minimize the cost sensitive multiclass hinge loss on f with a regularization term on the model parameters. Our objective is given as:

$$\min \frac{1}{2}\sum_{t=1}^{T} \|\mathbf{w}_t\|^2 + \frac{C}{N}\sum_{i=1}^{N} \max_y\{f(\mathbf{x}_i, y) - f(\mathbf{x}_i, y_i) + \mathcal{C}(y_i, y)\} \tag{7}$$

where C is a parameter trading off between the regularization and loss. One can use other types of norms on $\mathbf{w}_t$ (e.g., ℓ_1 for sparsity/feature selection) without changing the optimization procedure below.

4.2. Forward stage-wise optimization

Like boosting, we use a stage-wise optimization procedure to minimize (7). Specifically, we define:

$$f_t(\mathbf{x}, y) = \sum_{s \le t} h_s(\mathbf{x}, y) \tag{8}$$

Then $f_t(\mathbf{x}, y) = f_{t-1}(\mathbf{x}, y) + h_t(\mathbf{x}, y)$. At round t, we consider a single weak learner $h_t(\mathbf{x}, y)$ with $f_{t-1}(\mathbf{x}, y)$ fixed. We define the *margin residual* at t as

$$\rho_t(i, y) = \mathcal{C}(y_i, y) + f_{t-1}(\mathbf{x}_i, y) - f_{t-1}(\mathbf{x}_i, y_i) \tag{9}$$

Then the optimization problem at round t is:

$$\min \frac{1}{2}\|\mathbf{w}_t\|^2 + \frac{C}{N}\sum_{i=1}^{N} \max_y \{\rho_t(i, y) + h_t(\mathbf{x}_i, y) - h_t(\mathbf{x}_i, y_i)\} \tag{10}$$

It is worth noting the connection between the margin residual $\rho_t(i, y)$ and the distribution $D_t(i, y)$ over pairs of instances and labels in exponential loss based boosting algorithms. Both $\rho_t(i, y)$ and $D_t(i, y)$ have the property that for instance and label pair $(\mathbf{x}_i, y)$ with high error, their values are high and vice versa. This property makes the weak learner at the current round focus more on those instance/label pairs that are not well classified by the current model.

Recall that $h_t(\mathbf{x}, y) = \mu_t(y) g_t(\mathbf{x})$, which makes (10) non-convex. We optimize (10) in a block coordinate descent way. Specifically, we iterate over two steps until convergence: optimizing over μ_t given g_t and optimizing over g_t given μ_t.

4.2.1. Initialization

To initialize the iterative algorithm, we need a good starting point. Consider a pair of an instance and a label, i.e., (i, y). There are two cases: (1) if $\mu_t(y_i) = \mu_t(y)$, then $h_t(\mathbf{x}_i, y) = h_t(\mathbf{x}_i, y_i)$, which implies the margin residual stays the same as $\rho_t(i, y)$; (2) if $\mu_t(y_i) \neq \mu_t(y)$, then the new margin residual becomes $\rho_t(i, y) - 2\mu_t(y_i) g_t(\mathbf{x}_i)$ (since $\mu_t(y) = -\mu_t(y_i)$), so we can learn a good $g_t(\mathbf{x}_i)$ to drive the margin residual down. Based on this observation, it would be a nice goal to partition the classes such that high $\rho_t(i, y)$ can be pushed down. Formally, we want to maximize

$$\sum_{i=1}^{N} \sum_{y \in \mathcal{Y}} \rho_t(i, y) \mathbf{1}\{\mu_t(y_i) \neq \mu_t(y)\} \tag{11}$$

where $\mathbf{1}\{\cdot\}$ is an indicator function. After re-grouping the terms, we get:

$$\max \sum_{(y, y'): y < y'} \mathbf{1}\{\mu_t(y) \neq \mu_t(y')\} \cdot \left(\sum_{i: y_i = y} \rho_t(i, y') + \sum_{i: y_i = y'} \rho_t(i, y) \right) \tag{12}$$

(12) has an intuitive interpretation: for a pair of classes (y, y'), we to partition them into two different groups, if their inter-class margin residuals are high. This a max-cut problem which we solve as an SDP. Note that both Schapire (1997); Li (2006) found that a random initialization might give better performance due to a better tradeoff between the error-correcting property of the partition and the hardness of the induced binary problem. We also tried a version of the random initialization, where we randomly generate n μ_t's, and choose the one with the highest value of (11).

4.2.2. Optimizing the binary classifier

With μ_t fixed, we reformulate (10) as:

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{w}_t\|^2 + \frac{C}{N}\sum_{i=1}^{N} \xi_{t,i} \\ &\text{s.t. } h_t(\mathbf{x}_i, y_i) - h_t(\mathbf{x}_i, y) \ge \rho_t(i, y) - \xi_{t,i},\ \forall (i, y) \end{aligned} \tag{13}$$

Since $h_t(\mathbf{x}, y) = \mu_t(y) g_t(\mathbf{x})$, for each instance and label pair, there are two cases: (1) if $\mu_t(y_i) = \mu_t(y)$, then $h_t(\mathbf{x}_i, y_i) - h_t(\mathbf{x}_i, y) = 0$; (2) if $\mu_t(y_i) \neq \mu_t(y)$, then $h_t(\mathbf{x}_i, y_i) - h_t(\mathbf{x}_i, y) = 2\mu_t(y_i) g_t(\mathbf{x}_i)$. So we can reformulate the above problem as (assume $g_t(\mathbf{x}) = \mathbf{w}_t \phi(\mathbf{x}) + b_t$, absorb 2 into $\mathbf{w}_t, b_t$, and drop the subscript of t to avoid notation clutter):

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N} \xi_i \\ &\text{s.t. } \xi_i \ge \rho_t(i, y),\ \forall (i, y) : \mu(y) = \mu(y_i) \\ &\qquad \forall (i, y) : \mu(y) \neq \mu(y_i) \\ &\qquad \mu(y_i)(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \ge \rho_t(i, y) - \xi_i \end{aligned} \tag{14}$$

There are $N \times K$ constraints in (14), but not all of them are active. Denote $\rho_i^- = \max_{y: \mu(y) \neq \mu(y_i)} \rho_t(i, y)$ and $\rho_i^+ = \max_{y: \mu(y) = \mu(y_i)} \rho_t(i, y)$. Note that $\rho_i^+ \ge 0$, since $\rho(i, y_i) = 0$. We also define $\xi_i' = \xi_i - \rho_i^+$, then (14) is equivalent to (ignoring a constant in the objective):

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N} \xi_i' \\ &\text{s.t. } \mu(y_i)(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \ge (\rho_i^- - \rho_i^+) - \xi_i',\ \forall i \\ &\qquad \xi_i' \ge 0, \qquad \forall i \end{aligned} \tag{15}$$

Now, there are only $2N$ constraints, which is independent of the number of classes. This is a margin rescaling version of the binary SVM, and we modified the highly optimized LIBSVM (Chang & Lin, 2001) to solve it. The dual of (15) is as follows:

$$\max \sum_i \alpha_i(\rho_i^- - \rho_i^+) - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j\mu(y_i)\mu(y_j)\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) \quad (16)$$
$$\text{s.t.} \sum_i \alpha_i\mu(y_i) = 0 \quad \text{and} \quad 0 \le \alpha_i \le \frac{C}{N},\ \forall i$$

One can easily kernelize (16) by replacing $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ with a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$.

4.2.3. Optimizing the codebook column

The second step in our block coordinate descent procedure is to fix $g_t(\mathbf{x})$ and optimize $\mu_t(y)$. The intuition is that the initial coloring might induce a hard binary classification problem with high errors, and re-coloring based on the learned binary classifier may give better generalization error. Under the unified objective (10), with $g_t(\mathbf{x})$ fixed, the objective can be rewritten as:

$$\sum_{i=1}^N \max_y\{\rho_t(i,y) + \mu_t(y)g_t(\mathbf{x}_i)\} - \sum_{k=1}^K \mu_t(k)\sum_{i:y_i=k} g_t(\mathbf{x}_i) \quad (17)$$

Different $\mu_t(y)$'s are coupled by the *max* function. We use an ICM style descent procedure to get a local optimal of this combinatorial problem. Specifically, given some ordering of $\mu_t(y)$'s, we change the value of one $\mu_t(y)$ at a time, if the resulting objective value is smaller, then we take this change, and move to the next $\mu_t(y')$. This is repeated until no single change of $\mu_t(y)$ can decrease the objective. We found that due to the good initialization and strong $g_t(\mathbf{x})$, this procedure often converges after one or two iterations.

The HingeBoost.OC algorithm is summarized in Algorithm 1. At each round, we first solve (12) to get the initialization of μ_t, and then iterate over learning the binary classifier and the re-coloring until convergence or a pre-specified number of iterations.

5. Experiments

Basic Setup. In the experiments, we compare the performance of HingeBoost.OC to other popular methods based on reducing multiclass to binary: one-vs-one, one-vs-all, ECOC (Dietterich & Bakiri, 1995; Allwein et al., 2001), AdaBoost.ERP [1] (Li, 2006) (a representative multiclass boosting algorithm based on output coding and the sum-of-exponential loss), and the single-machine method: multiclass SVM (Crammer & Singer, 2002). For the ECOC method, the coding matrix is chosen before seeing the data such that the minimum Hamming distance between any pair of codewords is maximized (good error-correcting property). For a K-class problem, if $3 \le K \le 7$, we used the exhaustive codes (Dietterich & Bakiri, 1995) with the codeword length $2^{k-1}-1$. If $K > 7$, we used the dense random codes from (Allwein et al., 2001). Specifically, the random codes has $\lceil 10\log_2(K)\rceil$ columns. Each element in the coding matrix is chosen at random from $\{-1, +1\}$. We examined 10,000 such random coding matrices and chose the one that has the largest minimum pair-wise Hamming distances between all pairs of codewords and does not have any identical columns. Furthermore, we tried both Hamming decoding and exponential-loss-based decoding (Allwein et al., 2001), and found that the latter is consistently better, so we report ECOC performance using the loss-based decoding scheme.

For HingeBoost.OC, we fixed the maximum number of iterations of line 5-8 in Algorithm 1 to be 2, and we found that in most cases, iteratively optimiz-

[1] We used the code from the author to run experiments.

Algorithm 1 HingeBoost.OC

Input: A training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$; number of rounds T; regularization parameter C; a cost matrix $\mathcal{C}(y, y')$.

1: Initialize $\forall(i,y), \rho_1(i,y) = \mathcal{C}(y_i, y)$
2: **for** $t = 1$ to T **do**
3: Initialize the column $\mu_t \in \{-1,+1\}^K$ by maximizing (12).
4: **repeat**
5: $\forall i, \rho_i^- = \max_{y:\mu_t(y)\neq\mu_(y_i)} \rho_t(i,y)$
6: $\forall i, \rho_i^+ = \max_{y:\mu_t(y)=\mu_t(y_i)} \rho_t(i,y)$
7: Fix μ_t, solve (15) to obtain $g_t(\mathbf{x})$
8: Fix $g_t(\mathbf{x})$, solve (17) to obtain new μ_t
9: **until** convergence or some specified # of steps
10: $\forall(i,y)$, $\rho_{t+1}(i,y) = \rho_t(i,y) + \mu_t(y)g_t(\mathbf{x}_i) - \mu_t(y_i)g_t(\mathbf{x}_i)$
11: **end for**

Return: $M = [\mu_1, \dots \mu_T]$ and $\{g_t(\mathbf{x})\}_{t=1}^T$

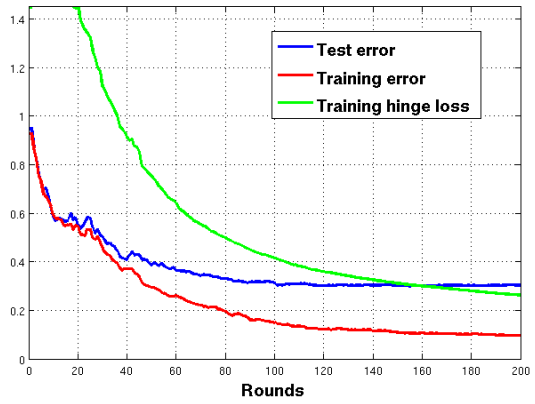

Figure 2. A typical set of learning curves on the scene15 dataset with GIST feature, showing the training/testing errors and the training hinge loss over rounds.

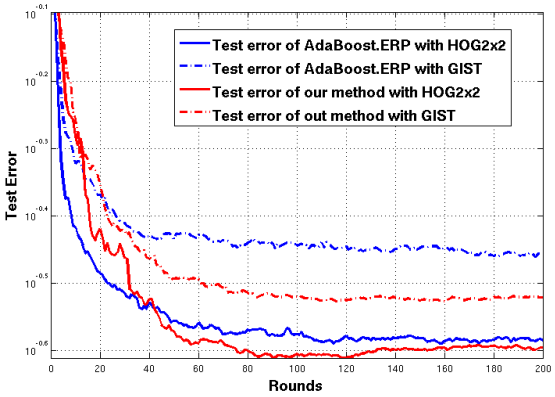

Figure 3. Comparison of test errors over rounds using AdaBoost.ERP and HingeBoost.OC on scene15 dataset based on GIST and HOG2x2 features.

ing the column and the binary classifiers converges in one or two steps due to the relatively strong weak learner, i.e., margin-rescaled SVM. In addition, for HingeBoost.OC, we tried two methods for the column initialization. One is to use SDP to solve (12) and the other is to randomly generate 1000 μ_t's and choose the one that gives the highest value of (11). We found empirically that the latter is usually better (in terms of the final classification performance) and faster, so we used it as the initialization method in the experiment. For all methods, the binary classifiers or weak learners are SVMs. The SVM parameter C is chosen by cross validation on the training set.

Scene15 Dataset. The first dataset we used is the scene15 dataset (Lazebnik et al., 2006). It is composed of 9 outdoor scenes (tall building, inside city, etc.) and 6 indoor scenes (office, kitchen, etc.). There are 4485 images in total, and 1500 are randomly sampled (100 per class) for training and the remaining 2985 for test. This is repeated 10 times, and the mean and standard deviation of the test classification errors are computed. We used two types of features: a weak one called GIST (Oliva & Torralba, 2001) and a strong one based on spatially distributed histogram of gradient, denoted as HOG2x2 (Xiao et al., 2010).

First, we study how the hinge loss and training/test errors change over rounds for HingeBoost.OC. As can be seen from Figure 2: (1) HingeBoost.OC can successfully drive down the hinge loss and training/test errors over rounds; (2) although the hinge loss and training errors keep going down even beyond 200 rounds, the test error converges around 100 rounds; (3) the test error stays flat for a large number of rounds, which suggests that HingeBoost.OC is robust to overfitting when the number of rounds is large.

Second, we show in Figure 3 the test classification errors over rounds for both AdaBoost.ERP and HingeBoost.OC. Given a weak feature, i.e., GIST, HingeBoost.OC achieves much better test errors than that of AdaBoost.ERP. However, if a strong feature is used (also meaning a stronger weak learner), this gap is reduced, but HingeBoost.OC still converges to a lower error. If the weak learner is not strong, initializing using a one-vs-all codebook helps the boosting algorithm converge to a better solution. For example, using GIST feature, the test errors of AdaBoost.ERP with and without one-vs-all codebook initialization are 35.54 ± 0.93 and 36.46 ± 1.27. For HingeBoost.OC, the test errors with and without one-vs-all codebook initialization are 29.35 ± 0.73 and 30.37 ± 0.66 respectively. In the experiments, we tried both configurations and report the better one.

Last, we show the test errors on scene15 dataset for different methods in Table 1. For both AdaBoost.ERP and HingeBoost.OC, we fix the number of rounds to be 200. As shown in Table 1, in cases of both weak and strong features, HingeBoost.OC achieves the lowest errors, but the performance gain is more significant when the weak feature is used. Note that the overlap between error bars are mainly due to inter-fold variations. For example, using the GIST feature, a paired t-test between the score vectors from one-vs-one and ours rejects the null hypothesis with the p value equal to 8×10^{-5}. It is worth noting that ECOC has the worst performance on average, although the minimum Hamming distance of the coding matrix used by ECOC is 32 compared to 2 of the one-vs-all codebook.

UCI Dataset. We further tested different methods on standard multiclass problems from the UCI machine learning repository (Newman et al., 1998). The five datasets we used are: pendigits (pen-based recognition of hand written digits), satimage (pixel-based classification on satellite images), segment (multiclass image segmentation), vehicle (object recognition) and vowel (speech recognition). Following the same protocol used by Li (2006), for datasets with both training and test sets, experiments were run 100 times and the results were averaged (due to the randomness factor in AdaBoost.ERP and HingeBoost.OC). Otherwise, a 10-fold cross-validation was repeated for 10 times for each fold with a total of 100 runs. As before, we fixed the number of rounds of both boosting algorithms to be 200. We first conducted the experiments using linear SVM as weak learner. The results are shown in Table 2. In this case, HingeBoost.OC achieved the best performance on two datasets while one-vs-one did the best on the other three. Note that on those three datasets where one-vs-one is the best, HingeBoost.OC is consistently better than all the other methods.

Recall that HingeBoost.OC can be naturally ker-

Table 1. Test errors (%) on scene15 dataset.

feature	one-vs-one	one-vs-all	ECOC	AdaBoost.ERP	Multiclass SVM	HingeBoost.OC
GIST	30.49 ± 1.00	32.16 ± 1.00	39.73 ± 0.16	35.54 ± 0.93	31.26 ± 0.50	$\mathbf{29.35 \pm 0.73}$
HOG2x2	24.24 ± 0.59	24.47 ± 0.68	28.03 ± 0.12	26.24 ± 0.55	26.08 ± 0.59	$\mathbf{24.18 \pm 0.66}$

Table 2. Test errors (%) on datasets from the UCI repository using linear SVM as base learner.

dataset	one-vs-one	one-vs-all	ECOC	AdaBoost.ERP	Multiclass SVM	HingeBoost.OC
pendigits	**4.61**	10.18	17.89	8.18 ± 0.35	8.46	6.95 ± 0.20
satimage	**14.22**	16.75	19.4	16.99 ± 0.09	16.15	16.05 ± 0.14
segment	5.84 ± 0.15	10.61 ± 0.23	8.33 ± 0.16	6.16 ± 0.16	5.37 ± 0.18	$\mathbf{5.11 \pm 0.15}$
vowel	**49.78**	58.44	68.61	57.60 ± 0.33	67.53	52.90 ± 0.92
vehicle	21.40 ± 0.43	21.13 ± 0.44	21.16 ± 0.38	22.36 ± 0.40	21.43 ± 0.51	$\mathbf{20.06 \pm 0.40}$

nelized as shown in (16), so we also conducted experiments on the UCI datasets using perceptron kernel (Lin & Li, 2005). The choice of the kernel is for direct comparison to the results reported in (Li, 2006) (we also used the same fold partition generated by the code from the author). Note that all binary classifiers and weaker learners used by other methods are using the same kernel. For multiclass SVM, we found empirically that this type of kernel does not work well in the multiclass setting, so we used an RBF kernel which is shown comparable to the perceptron kernel in (Lin & Li, 2005). The γ parameter in the RBF kernel is chosen from $\{0.1, 0.5, 1, 2, 4, 8, 16\}$ by cross validation. The test errors are shown in Table 3 (the first and the third columns are from (Li, 2006)). First, with non-linear kernels, all methods achieved better performance compared to linear kernel. Second, with strong binary classifier, it seems that no method is significantly better than others on all datasets. HingeBoost.OC achieved the lowest errors on two datasets while AdaBoost.ERP, one-vs-one and multiclass SVM performed the best on the other three respectively. AdaBoost.ERP and HingeBoost.OC did best on harder datasets (the three with the highest errors). Again, it is worth noting that with strong binary classifier, the performance gap between ECOC and other methods are greatly reduced.

SUN Dataset. Finally, we conducted experiments on a large-scale dataset, the SUN dataset (Xiao et al., 2010). The SUN dataset is by far the largest scene recognition dataset, and it contains 899 categories. We used 397 well-sampled categories to run an experiment. This subset contains $39,700$ images. We used 10-fold cross validation to test different methods. For each fold, each class has 50 training images and 50 testing images. We used the strong feature HOG2x2 with linear SVM as the base learner. Since the AdaBoost.ERP code from (Li, 2006) does not support sparse data structure (dense data requires around 8GB memory) and also it runs very slowly due to parameter searching for each weak learner at each round, we do not have the result of AdaBoost.ERP on this dataset. For HingeBoost.OC, we fixed the number of rounds to be 1000, and used the one-vs-all codebook as initialization. The results are shown in Table 4. First, on such a large and difficult dataset, the test errors of all methods are high. Compared to the error of random guessing 99.75% and the error of using GIST feature with RBF kernel 83.7% in (Xiao et al., 2010), our results are reasonable. Second, different algorithms seem to perform similarly, but HingeBoost.OC and multiclass SVM are slightly better. The overlap between the error bars of one-vs-one and ours is mainly due to inter-fold variations. A paired t-test between the score vectors from one-vs-one and ours rejects the null hypothesis with the p value equal to 0.019.

Discussion. There are two metrics to evaluate a classification algorithm. The first one is the accuracy. In our experiments, HingeBoost.OC achieved the best classification accuracy on 7 cases out of 13 runs, and one-vs-one is the best on 4 of them. However, in terms of the testing speed, one-vs-one does not scale well due to the quadratical growth of the complexity in the number of classes. For example, on the SUN dataset, one-vs-one requires $78,606$ classifier evaluations while HingeBoost.OC only needs 1000 or less. Furthermore, in case of linear weak learner, HingeBoost.OC can be further reduced to K "classifier" evaluations for a K-class problem, since

$$\begin{aligned} f(\mathbf{x}, y) &= \textstyle\sum_{t=1}^{T} \mu_t(y) \mathbf{w}_t^T \phi(\mathbf{x}) + \mu_t(y) b_t \\ &= (\textstyle\sum_t \mu_t(y) \mathbf{w}_t)^T \phi(\mathbf{x}) + \sum_t \mu_t(y) b_t \end{aligned} \tag{18}$$

where $\sum_t \mu_t(y)\mathbf{w}_t$ and $\sum_t \mu_t(y) b_t$ can be precomputed, making the test time independent of the number of rounds T. This implies that HingeBoost.OC is as efficient as one-vs-all in the linear case, but is much more accurate.

It is worth noting the comparison between the multiclass hinge loss based methods, i.e., multiclass SVM and HingeBoost.OC, and the sum-of-exponential loss based method, i.e., AdaBoost.ERP. In our experiments, multiclass SVM performs better than AdaBoost.ERP on 7 out of 12 cases, and HingeBoost.OC

Table 3. Test errors (%) on datasets from the UCI repository using SVM with perceptron kernel as base learner.

dataset	min(one-vs-one,one-vs-all)	ECOC	AdaBoost.ERP	Multiclass SVM	HingeBoost.OC
pendigits	1.71	1.83	1.64 ± 0.02	**1.55**	1.83 ± 0.03
satimage	7.70	7.70	7.72 ± 0.05	7.85	$\mathbf{7.40 \pm 0.20}$
segment	$\mathbf{2.09 \pm 0.09}$	2.23 ± 0.10	2.16 ± 0.09	2.81 ± 0.11	2.39 ± 0.11
vowel	37.45	43.72	39.95 ± 0.19	35.30	$\mathbf{34.85 \pm 0.93}$
vehicle	17.89 ± 0.37	18.79 ± 0.40	$\mathbf{17.67 \pm 0.35}$	20.60 ± 0.32	18.30 ± 0.58

Table 4. Test errors (%) on the SUN dataset.

feature	one-vs-one	one-vs-all	ECOC	Multiclass SVM	HingeBoost.OC
HOG2x2	82.41 ± 0.23	82.72 ± 0.32	93.56 ± 0.17	$\mathbf{82.24 \pm 0.38}$	$\mathbf{82.16 \pm 0.14}$

is better than AdaBoost.ERP on 9 out of 12 cases. These empirical results are consistent with the analysis of these two loss functions in Section 2.

It is also interesting to note the connections between multiclass SVM and HingeBoost.OC. Both methods are trying to find K hyperplanes in the feature space explicitly or implicitly represented by the mapping $\phi(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}^D$. (18) shows why this is the case for HingeBoost.OC. Although both methods use the same loss, the regularization and the optimization procedure are different. Empirically the stage-wise optimization and regularization seems to give better performance, which might be due to lower variance than that of the single joint optimization by multiclass SVM.

Finally, it would be interesting to combine HingeBoost.OC with one-vs-one by constructing a pool of candidate columns based on all-pairs of classes. At each round, HingeBoost.OC can choose a column from this pool which will induce an easy binary problem with good statistical performance.

6. Conclusion and Future Work

Multiclass classification problems are very natural and important in many domains. Designing a multiclass classification algorithm that can scale up to hundreds of classes in terms of both high accuracy and low computational complexity is very challenging. Future works towards this direction include: (1) introducing 0 value in the partition to achieve better accuracy (ignoring some confusing classes will lead to an easier binary problem with better performance); (2) a dynamic decision making scheme based on margins achieved by evaluating a partial number of weak learners to avoid going through all stages for all instances.

Acknowledgment. This work was supported by the NSF under grant No. RI-0917151 and MURI contract N000140710747 and N00014-10-10933. We thank Yoram Singer, Koby Crammer, Cliff Chiung-Yu Lin and the reviewers for helpful comments.

References

Allwein, E. L., Schapire, R. E., and Singer, Y. Reducing multiclass to binary: a unifying approach for margin classifiers. *J. Mach. Learn. Res.*, 1:113–141, 2001.

Amit, Y., Fink, M., Srebro, N., and Ullman, S. Uncovering shared structures in multiclass classification. In *ICML*, 2007.

Chang, C.C. and Lin, C.J. *LIBSVM: a library for support vector machines*, 2001.

Crammer, K. and Singer, Y. On the learnability and design of output codes for multiclass problems. In *In Proc. of the 13th Annual Conf. on COLT*, 2000.

Crammer, K. and Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2002.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Dietterich, T. G. and Bakiri, G. Solving multiclass learning problems via error-correcting output codes. *J. of A. I. Res.*, 2:263–286, 1995.

Guruswami, V. and Sahai, A. Multiclass learning, boosting, and error-correcting codes. In *Proc. of the Twelfth Annual Conf. on Computational Learning Theory*, 1999.

Hastie, T., Tibshirani, R., and Friedman, J. H. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR06*, 2006.

Li, Ling. Multiclass boosting with repartitioning. In *In: Proc. 23rd Int. Conf. Machine Learning*, 2006.

Lin, H.T. and Li, L. Novel distance-based SVM kernels for infinite ensemble learning. In *ICNIP*, 2005.

Newman, D.J., Hettich, S., Blake, C.L., and Merz, C.J. Uci repository of machine learning databases, 1998.

Oliva, A. and Torralba, A. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, 42:145–175, 2001.

Rätsch, G., Smola, A.J., and Mika, S. Adapting codes and embeddings for polychotomies. In *NIPS*, 2003.

Schapire, R.E. Using output codes to boost multiclass learing problems. In *ICML*, 1997.

Sun, Y., Todorovic, S., Li, J., and Wu, D. Unifying the error-correcting and output-code adaboost within the margin framework. In *In: Proc. 22rd ICML*, 2005.

Xiao, J.X., Hays, J., Ehinger, K.A., Oliva, A., and Torralba, A.B. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR10*, 2010.

Zou, H., Zhu, J., and Hastie, T. New multicategory boosting algorithms based on multicategory fisher-consistent losses. *Annals of Applied Statistics*, 2:1290–1306, 2008.