Totally Corrective Multiclass Boosting with Binary Weak Learners

Zhihui Hao, Chunhua Shen, Nick Barnes, and Bo Wang

***Abstract*—In this work, we propose a new optimization framework for multiclass boosting learning. In the literature, AdaBoost.MO and AdaBoost.ECC are the two successful multiclass boosting algorithms, which can use binary weak learners. We explicitly derive these two algorithms' Lagrange dual problems based on their regularized loss functions. We show that the Lagrange dual formulations enable us to design totally-corrective multiclass algorithms by using the primal-dual optimization technique. Experiments on benchmark data sets suggest that our multiclass boosting can achieve a comparable generalization capability with state-of-the-art, but the convergence speed is much faster than stage-wise gradient descent boosting. In other words, the new totally corrective algorithms can maximize the margin more aggressively.**

***Index Terms*—Multiclass boosting, totally corrective boosting, column generation, convex optimization.**

I. Introduction

Boosting is a powerful learning technique for improving the accuracy of any given classification algorithm. It has been attracting much research interest in the machine learning and pattern recognition community. Since Viola and Jones applied boosting to face detection [1], it has shown great success in computer vision, including the applications of object detection and tracking [2], [3] generic object recognition [4], image classification [5] and retrieval [6].

The essential idea of boosting is to find a combination of *weak* hypotheses generated by a base learning oracle. The learned ensemble is called the *strong* classifier in the sense that it often achieves a much higher accuracy. One of the most popular boosting algorithms is AdaBoost [7], which has been proven a method of minimizing the regularized exponential loss function [7], [8]. There are many variations on AdaBoost in the literature. For example, LogitBoost [9], optimizes the logistic regression loss instead of the exponential loss. To understand how boosting works, Schapire *et al.* [10] introduced the margin theory and suggested that boosting is especially effective at maximizing the margins of training examples. Based on this concept, Demiriz *et al.* [11] proposed LPBoost, which maximizes the minimum margin using the hinge loss.

Since most of the pattern classification problems in real world are multiclass problems, researchers have extended binary boosting algorithms to the multiclass case. For example in [12], Freund and Schapire have described two possible extensions of AdaBoost to the multiclass case. AdaBoost.M1 is the first and perhaps the most direct extension. In AdaBoost.M1, a weak hypothesis assigns only one of C possible labels to each instance. Consequently, the requirement for weak hypotheses that training error must be less than $1/2$ becomes harder to achieve, since random guessing only has an accuracy rate of $1/C$ in multiclass case. To overcome this difficulty, AdaBoost.M2 introduced a relaxed error measurement termed *pseudo-loss*. In AdaBoost.M2, the weak hypothesis is required to answer $C-1$ questions for one training example $(\boldsymbol{x}_i, y_i)$: which is the label of $\boldsymbol{x}_i$, c or y_i $(\forall c \neq y_i)$? A falsely matched pair $(\boldsymbol{x}_i, c)$ is called a *mislabel*. Pseudo-loss is defined as the weighted average of the probabilities of all incorrect answers. Recently, Zhu *et al.* [13] proposed a multiclass exponential loss function. Boosting algorithms based on this loss, including SAMME [13] and GAMBLE [14] only require the weak hypothesis performs better than random guessing $(1/C)$.

The above-mentioned multiclass boosting algorithms have a common property: the employed weak hypotheses should have the ability to give predictions on all C possible labels at each call. Some powerful weak learning methods may be competent, like decision trees. However, they are complicated and time-consuming for training compared with binary learners. A higher complexity of assembled classifier often implies a larger risk of over-fitting the training data and possible decreasing of the generalization ability.

Therefore, it is natural to put forward another idea: if a multiclass problem can be reduced into multiple two-class ones, binary weak learning method such as linear discriminant analysis [15], [16], decision stump or product of decision stumps [17] might be applicable to these decomposed sub-problems. To make the reduction, one has to introduce some appropriate coding strategy to translate each label to a fixed binary string, which is usually referred to as a *codeword*. Then weak hypotheses can be trained at every bit position. For a test example, the label is predicted by decoding the codeword computed from the strong classifier. AdaBoost.MO [7] is a representative algorithm with this coding-decoding process. To increase the distance between codewords and thus improve the error correcting ability, Dietterich and Bakiri's error-correcting output codes (ECOC) [18] can be used in AdaBoost.MO. A

Z. Hao and B. Wang are with Beijing Institute of Technology, Beijing 100081, China. (e-mail: hzhbit@gmail.com and wangbo@bit.edu.cn), Z. Hao's contribution was made when he was visiting NICTA Canberra Research Laboratory and the Australian National University.

C. Shen and N. Barnes are with NICTA, Canberra Research Laboratory, Canberra, ACT 2601, Australia, and also with the Australian National University, Canberra, ACT 0200, Australia (e-mail: chunhua.shen@nicta.com.au, nick.barnes@nicta.com.au). Correspondence should be addressed to C. Shen.

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Center of Excellence program. This work was also supported in part by the Australian Research Council through its special research initiative in bionic vision science and technology grant to Bionic Vision Australia.

variant of AdaBoost.MO, AdaBoost.OC, also combines boosting and ECOC. However, unlike AdaBoost.MO, AdaBoost.OC employs a collection of randomly generated codewords. For more details about the random methods, we refer the reader to [19]. AdaBoost.OC penalizes both the wrongly classified examples and the mislabels in correctly classified examples by calculating pseudo-loss. Hence, AdaBoost.OC may be viewed as a special case of AdaBoost.M2. The difference is that, weak hypotheses in AdaBoost.OC are required to answer one binary question for each training example: which is the label for $\boldsymbol{x}_i$ in the current round, 0 or 1? Later, Guruswami and Sahai [20] proposed a variant, AdaBoost.ECC, which replaces pseudo-loss with the common measurement to compute training errors.

In this work, we mainly focus on the multiclass boosting algorithms with *binary weak learners*. Specifically, AdaBoost.MO and AdaBoost.ECC. It has been proven AdaBoost.OC is in fact a shrinkage version of AdaBoost.ECC [21]. These two algorithms both perform stage-wise functional gradient descent procedures on the exponential loss function [21]. In [8], Shen and Li have shown that ℓ_1 norm regularized boosting algorithms including AdaBoost, LogitBoost and LPBoost, might be optimized through optimizing their corresponding dual problems. This primal-dual optimization technique is also *implicitly* applied by other studies to explore the principles of boosting learning [22], [23]. Here we study AdaBoost.MO and AdaBoost.ECC and explicitly derive their Lagrange dual problems. Based on the primal-dual pairs, we put these two algorithms into a column generation based primal-dual optimization framework. We also analytically show the boosting algorithms that we proposed are *totally corrective* in a relaxed fashion. Therefore, the proposed algorithms seem to converge more effectively and maximize the margin of training examples more aggressively. To our knowledge, our proposed algorithms are the first totally corrective multiclass boosting algorithms.

The notation used in this paper is as follows. We use the symbol $\mathbf{M}$ to denote a coding matrix with $M(a,b)$ being its (a,b)-th entry. Bold letters $(\boldsymbol{u},\boldsymbol{v})$ denote column vectors. $\mathbf{0}$ and $\mathbf{1}$ are vectors with all entries being 0 and 1 respectively. The inner product of two column vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are expressed as $\boldsymbol{u}^\top \boldsymbol{v} = \sum_i u_i v_i$. Symbols $\succeq$, $\preceq$ placed between two vectors indicate that the inequality relationship holds for all pairwise entries. Double-barred letters ($\mathbb{R}$, $\mathbb{Y}$) denote specific domains or sets. The abbreviation s.t. means "subject to". $\mathbb{1}(\pi)$ denotes an indicator function which gives 1 if π is true and 0 otherwise.

The remaining content is organized as follows. In Section II we briefly review the coding strategies in multiclass boosting learning and describe the algorithms of AdaBoost.MO and AdaBoost.ECC. Then in Section III we derive the primal-dual relations and propose our multiclass boosting learning framework. In Section IV we compare the related algorithms through several experiments. We conclude the paper in Section V.

II. Multiclass boosting algorithms and coding matrix

In this section, we briefly review the multiclass boosting algorithms of AdaBoost.MO [7] and AdaBoost.ECC [20].

Algorithm 1 AdaBoost.MO (Schapire and Singer, 1999)

```
Input training data $(\boldsymbol{x}_i, y_i)$, $y_i \in \{1,\dots,C\}$, $i = 1,\dots,N$;
        maximum training iterations $T$, and the coding matrix $\mathbf{M}^{C\times L}$.
Initialization
    Weight distribution
    $u_{i,l} = \frac{1}{NL}$, $i = 1,\dots,N$, $l = 1,\dots,L$.
for $t = 1 : T$ do
    a) Normalize $\boldsymbol{u}$;
    b) Train $L$ weak hypotheses $h_l^{(t)}(\cdot)$ according to the weight distribution $\boldsymbol{u}$;
    c) Compute $\epsilon = \sum_i \sum_l u_{i,l} \mathbb{1}(M(y_i,l) \neq h_l^{(t)}(\boldsymbol{x}_i))$;
    d) Compute $\omega^{(t)} = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$;
    e) Update $u_{i,l} = u_{i,l}\exp\big(-\omega^{(t)} M(y_i,l) h_l^{(t)}(\boldsymbol{x}_i)\big)$;
end for
Output $\boldsymbol{f}(\cdot) = \big[\sum_t \omega^{(t)} h_1^{(t)}(\cdot), \cdots, \sum_t \omega^{(t)} h_L^{(t)}(\cdot)\big]^\top$.
```

A typical multiclass classification problem can be expressed as follows. A training set for learning is given by $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$. Here $\boldsymbol{x}_i$ is a pattern and y_i is the label, which takes a value from the space $\mathbb{Y} = \{1,2,\dots,C\}$ if we have C classes. The goal of classification is then to find a classifier $\boldsymbol{f}: \mathbb{X} \to \mathbb{Y}$ which assigns one and only one label to a new observation $(\boldsymbol{x}, y)$ with a minimal probability of $y \neq \boldsymbol{f}(\boldsymbol{x})$. Boosting algorithm tries to find an ensemble function in the form of $\boldsymbol{f}(\boldsymbol{x}) = \sum_{t=1}^T \omega^{(t)} h^{(t)}(\boldsymbol{x})$ (or equivalently the normalized version $\sum_{t=1}^T \boldsymbol{f}(\boldsymbol{x}) / \sum_t \omega^{(t)}$), where $h(\cdot)$ denotes the weak hypotheses generated by base learning algorithm and $\boldsymbol{\omega} = [\omega^{(1)} \cdots \omega^{(T)}]^\top$ denotes the associated coefficient vector. Typically, a weight distribution $\boldsymbol{u}$ is used on training data, which essentially makes the learning algorithm concentrate on those examples that are hard to distinguish. The weighted training error of hypothesis $h(\cdot)$ on $\boldsymbol{u}$ is given by $\sum_i u_i \mathbb{1}(y_i \neq h(\boldsymbol{x}_i))$.

To decompose a multiclass problem into several binary subproblems, a coding matrix $\mathbf{M} \in \{\pm 1\}^{C\times L}$ is required. Let $M(c,:)$ denote the c-th row, which represents a L-length codeword for class c. One binary hypothesis can be learned then for each column, where training examples has been relabeled into two classes. For a newly observed instance $\boldsymbol{x}$, $\boldsymbol{f}(\boldsymbol{x})$ outputs an unknown codeword. Hamming distance or some loss-based measure is used to calculate the distances between this word and rows in $\mathbf{M}$. The "closest" row is identified as the predicted label. For binary strings, loss-based measures are equivalent to Hamming distance.

The coding process can be viewed as a mapping to a new higher dimensional space. If the codewords in the new space are mutually distant, the more powerful error-correction ability can be gained. For this reason, Dietterich and Bakiri's error-correcting output codes (ECOC) are usually chosen. Especially if the coding matrix equals to the unit matrix (*i.e.* , each codeword is one basis vector in the high dimensional space), no error code could be corrected. This is the *one-against-all* or *one-per-class* approach. Several coding matrices have been evaluated in [24]. Another family of codes are random codes [19], [20], [25]. Compared with fixed codes, random codes

Algorithm 2 AdaBoost.ECC (Guruswami and Sahai, 1999)

```
Input training data $(\boldsymbol{x}_i, y_i)$, $y_i \in \{1, \dots, C\}$, $i = 1, \dots, N$;
      maximum training iterations $T$.
Initialization
    The coding matrix $\mathbf{M} = [\,]$, and the weight distribution
    $u_{i,c} = \frac{1}{N(C-1)}$, $i = 1, \dots, N$, $c = 1, \dots, C$.
for $t = 1 : T$ do
  a) Create $M(:, t) \in \{-1, +1\}^{C \times 1}$;
  b) Normalize $\boldsymbol{u}$;
  c) Compute weight distribution for mislabels
       $d_i = \sum_c u_{i,c} \mathbb{1}(M(c,t) \neq M(y_i, t))$;
  d) Normalize $\boldsymbol{d}$;
  e) Train a weak hypothesis $h^{(t)}(\cdot)$ using $\boldsymbol{d}$;
  f) Compute $\epsilon = \sum_i d_i \mathbb{1}(M(y_i, t) \neq h^{(t)}(\boldsymbol{x}_i))$;
  g) Compute $\omega^{(t)} = \frac{1}{4} \ln(\frac{1-\epsilon}{\epsilon})$;
  h) Update $u_{i,c} =$
       $u_{i,c} \exp\left(-\omega^{(t)} \left(M(y_i, t) - M(c, t)\right) h^{(t)}(\boldsymbol{x}_i)\right)$;
end for
Output $\boldsymbol{f}(\cdot) = [\omega^{(1)} h^{(1)}(\cdot), \cdots, \omega^{(T)} h^{(T)}(\cdot)]^\top$.
```

are more flexible to explore the relationships between different classes.

AdaBoost.MO employs a predefined coding matrix, such as ECOC. An L-dimensional weak hypothesis is trained at each iteration, with one entry for a binary subproblem. The pseudo-code of AdaBoost.MO is given in Algorithm 1. For a new observation, the label can be predicted by

$$\begin{aligned} y &= \arg\max_{c=1}^{C} \{M(c,:)\boldsymbol{f}(\boldsymbol{x})\} \\ &= \arg\max_{c} \{\sum_{j=1}^{T} \omega^{(j)} M(c,:) \boldsymbol{h}^{(j)}(\boldsymbol{x})\}. \end{aligned} \tag{1}$$

For AdaBoost.ECC, an incremental coding matrix is involved. At each iteration, a randomly generated code is added into the matrix, which corresponds a new binary subproblem being created. AdaBoost.ECC is summarized in Algorithm 2. The prediction is implemented by

$$y = \arg\max_{c} \{\sum_{j=1}^{T} \omega^{(j)} M(c,j) h^{(j)}(\boldsymbol{x})\}. \tag{2}$$

Comparing (2) with (1), we can see AdaBoost.ECC is similar to AdaBoost.MO in terms of coding-decoding process. Roughly speaking, the coding matrix in ECC is degenerated into a random changing column; accordingly, induced binary problems turn to be a single one. The relationship between these two algorithms will be completely clear after we derive the dual problems in the next section.

III. TOTALLY CORRECTIVE MULTICLASS BOOSTING

In this section we present the ℓ_1 norm regularized optimization problems that AdaBoost.MO and AdaBoost.ECC solve, and derive the corresponding Lagrange dual problems. Based on the column generation technique, we design new boosting methods for multiclass learning. Unlike conventional boosting, the proposed algorithms are totally corrective. For ease of exposition, we name our algorithms as MultiBoost.MO and MultiBoost.ECC.

A. Lagrange Dual of AdaBoost.MO

The loss function that AdaBoost.MO optimizes is proposed in [21]. Given a coding strategy $\mathbb{Y} \to \mathbf{M}^{C \times L}$, the loss function can be written as

$$L_{\text{MO}} = \sum_{i=1}^{N} \sum_{l=1}^{L} \exp\left(-M(y_i, l) F_l(x_i)\right), \tag{3}$$

where $F_l(x_i)$ is the l-th entry in the strong classifier and $F_l(x_i) = \sum_{t=1}^{T} \omega^{(t)} h_l^{(t)}(x_i)$. For a training example $\boldsymbol{x}_i$, let $\mathbf{H}_l(\boldsymbol{x}_i) = [h_l^{(1)}(\boldsymbol{x}_i)\ h_l^{(2)}(\boldsymbol{x}_i) \cdots h_l^{(T)}(\boldsymbol{x}_i)]^\top$ denote the outputs of the l-th weak hypothesis at all T iterations. The boosting process of AdaBoost.MO is equivalent to solve the following optimization problem:

$$\min_{\boldsymbol{\omega}} \quad \sum_{i=1}^{N} \sum_{l=1}^{L} \exp\left(-M(y_i, l) \mathbf{H}_l^\top(\boldsymbol{x}_i) \boldsymbol{\omega}\right) \tag{4}$$

$$\text{s.t.} \quad \boldsymbol{\omega} \succeq \mathbf{0}, \|\boldsymbol{\omega}\|_1 = \theta. \tag{5}$$

Notice that the constraint $\|\boldsymbol{\omega}\|_1 = \theta$ with $\theta > 0$ is not explicitly enforced in AdaBoost.MO. However, if the variable $\boldsymbol{\omega}$ is not bounded, one can arbitrarily make the loss function approach zero via enlarging $\boldsymbol{\omega}$ by an adequately large factor. For a convex and monotonically increasing function, $\|\boldsymbol{\omega}\|_1 = \theta$ is actually equivalent to $\|\boldsymbol{\omega}\|_1 \leq \theta$ since $\boldsymbol{\omega}$ always locates at the boundary of the feasibility set.

By using Lagrange multipliers, we are able to derive the Lagrange dual problem of any optimization problem [26]. If the strong duality holds, the optimal dual value is exactly the same as the optimal value of primal problem.

Theorem 1: The Lagrange dual problem of program (4) is

$$\begin{aligned} \max_{r, \boldsymbol{u}} \quad & -r\theta - \sum_{i=1}^{N} \sum_{l=1}^{L} u_{i,l} \log u_{i,l} + \mathbf{1}^\top \boldsymbol{u} \\ \text{s.t.} \quad & \sum_{i=1}^{N} \sum_{l=1}^{L} u_{i,l} M(y_i, l) \mathbf{H}_l^\top(\boldsymbol{x}_i) \preceq r \mathbf{1}^\top, \boldsymbol{u} \succeq \mathbf{0}. \end{aligned} \tag{6}$$

Proof: To derive this Lagrange dual, one needs to introduce a set of auxiliary variables $\gamma_{i,l} = -M(y_i, l) \mathbf{H}_l^\top(\boldsymbol{x}_i) \boldsymbol{\omega}$, $i = 1, \dots, N$, $l = 1, \dots, L$. Then we can rewrite the primal (4) into

$$\begin{aligned} \min_{\boldsymbol{\omega}} \quad & \sum_{i=1}^{N} \sum_{l=1}^{L} \exp \gamma_{i,l} \\ \text{s.t.} \quad & \gamma_{i,l} = -M(y_i, l) \mathbf{H}_l^\top(\boldsymbol{x}_i) \boldsymbol{\omega}, \\ & \boldsymbol{\omega} \succeq \mathbf{0}, \|\boldsymbol{\omega}\|_1 = \theta. \end{aligned} \tag{7}$$

The Lagrangian of the above program is

$$\begin{aligned} L(\boldsymbol{\omega}, \boldsymbol{\gamma}, \boldsymbol{q}, \boldsymbol{u}, r) = & \sum_{i=1}^{N} \sum_{l=1}^{L} \exp \gamma_{i,l} - \boldsymbol{q}^\top \boldsymbol{\omega} + r(\mathbf{1}^\top \boldsymbol{\omega} - \theta) \\ & - \sum_{i=1}^{N} \sum_{l=1}^{L} u_{i,k} \left(\gamma_{i,k} + M(y_i, l) \mathbf{H}_l^\top(\boldsymbol{x}_i) \boldsymbol{\omega}\right) \end{aligned} \tag{8}$$

with $\boldsymbol{q} \succeq \mathbf{0}$. The Lagrange dual function is defined as the infimum value of the Lagrangian over variables $\boldsymbol{\omega}$ and $\boldsymbol{\gamma}$.

$$\begin{aligned}\inf_{\boldsymbol{\omega},\boldsymbol{\gamma}} L &= \inf_{\boldsymbol{\omega},\boldsymbol{\gamma}} \sum_{i=1}^{N}\sum_{l=1}^{L}\exp\gamma_{i,l} - \sum_{i=1}^{N}\sum_{l=1}^{L}u_{i,l}\gamma_{i,l} - r\theta \\ &\quad -\overbrace{\Big(\sum_{i=1}^{N}\sum_{l=1}^{L}u_{i,l}M(y_i,l)\mathbf{H}_l^\top(\boldsymbol{x}_i) + \boldsymbol{q}^\top - r\mathbf{1}^\top\Big)}^{\text{must be } \mathbf{0}}\boldsymbol{\omega} \\ &= \inf_{\boldsymbol{\gamma}} \sum_{i=1}^{N}\sum_{l=1}^{L}\exp\gamma_{i,l} - \sum_{i=1}^{N}\sum_{l=1}^{L}u_{i,l}\gamma_{i,l} - r\theta \qquad (9) \\ &= -\sum_{i=1}^{N}\sum_{l=1}^{L}\overbrace{\sup_{\gamma}(u_{i,l}\gamma_{i,l} - \exp\gamma_{i,l})}^{\text{conjugate of exp. function}} - r\theta \\ &= -\sum_{i=1}^{N}\sum_{l=1}^{L}(u_{i,l}\log u_{i,l} - u_{i,l}) - r\theta. \qquad (10)\end{aligned}$$

The convex conjugate (or Fenchel duality) of a function $f : \mathbb{R} \to \mathbb{R}$ is defined as

$$g(y) = \sup_{x\in\mathbf{dom}f}(y^\top x - f(x)). \qquad (11)$$

Here we have used the fact that the conjugate of the exponential function $f(\gamma) = e^\gamma$ is $g(u) = u\log u - u$, if and only if $u \geq 0$. $0\log 0$ is interpreted as 0.

For each pair $(\boldsymbol{u}, r)$, the dual function gives a lower bound on the optimal value of the primal problem (4). Through maximizing the dual function, the best bound can be obtained. This is exactly the dual problem we derived. After eliminating $\boldsymbol{q}$ and collecting all the constraints, we complete the proof. ■

Since the primal (4) is a convex problem and satisfies Slater's condition [26], the strong duality holds, which means maximizing the cost function in (6) over dual variables $\boldsymbol{u}$ and r is equivalent to solve the problem (4). Then if a dual optimal solution $(\boldsymbol{u}^*, r^*)$ is known, any primal optimal point is also a minimizer of $L(\boldsymbol{\omega}, \boldsymbol{\gamma}, \boldsymbol{q}, \boldsymbol{u}^*, r^*)$. In other words, if the primal optimal solution $\boldsymbol{\omega}^*$ exists, it can be obtained by minimizing $L(\boldsymbol{\omega}, \boldsymbol{\gamma}, \boldsymbol{q}, \boldsymbol{u}^*, r^*)$ of the following function (see (9)):

$$\begin{aligned}-r^*\theta + \sum_{i=1}^{N}\sum_{l=1}^{L}\Big(&\exp\big(-M(y_i,l)\mathbf{H}_l^\top(\boldsymbol{x}_i)\boldsymbol{\omega}\big) \\ &- u_{i,l}^*\big(-M(y_i,l)\mathbf{H}_l^\top(\boldsymbol{x}_i)\boldsymbol{\omega}\big)\Big).\end{aligned} \qquad (12)$$

We can also use KKT conditions to establish the relationship between the primal variables and dual variables [26].

B. MultiBoost.MO: Totally Corrective Boosting based on Column Generation

Clearly, we can not obtain the optimal solution of (6) until all the weak hypotheses in constraints become available. In order to solve this optimization problem, we use an optimization technique termed column generation [11]. The concept of column generation is adding one constraint at a time to the dual problem until an optimal solution is identified. In our case, we find the weak classifier at each iteration that most violates the constraint in the dual. For (6), such a multidimensional weak classifier $\boldsymbol{h}^*(\cdot) = [h_1^*(\cdot)\cdots h_L^*(\cdot)]^\top$ can be found by solving the following problem:

$$\boldsymbol{h}^*(\cdot) = \operatorname*{argmax}_{\boldsymbol{h}(\cdot)} \sum_{l=1}^{L}\sum_{i=1}^{N}u_{i,l}M(y_i,l)h_l(\boldsymbol{x}_i), \qquad (13)$$

which is equivalent to solve L subproblems:

$$h_l^*(\cdot) = \operatorname*{argmax}_{h_l(\cdot)} \sum_{i=1}^{N}u_{i,l}M(y_i,l)h_l(\boldsymbol{x}_i),\ k = 1,\ldots,L. \qquad (14)$$

If we view $u_{i,l}$ as the weight of the coded training example $(\boldsymbol{x}_i, M(y_i,l))$, $i = 1,\ldots,N$, $l = 1,\ldots,L$, this is exactly the same as the strategy AdaBoost.MO uses. That is, to find L weak classifiers that produce the smallest weighted training error (maximum algebraic sum of weights of correctly classified data) with respect to the current weight distribution $\boldsymbol{u}$.

When a new constraint is added into the dual program, the optimal value of this maximization problem (6) would decrease. Accordingly the primal optimal value decreases too because of the zero duality gap. The primal problem is convex, which assures that our optimization problem will converges to the global extremum. In practice, MultiBoost.MO converges quickly on our tested datasets.

Next we need to find the connection between the primal variables $\boldsymbol{\omega}$ and dual variables $\boldsymbol{u}$ and r. According to KKT conditions, since the primal optimal $\boldsymbol{\omega}^*$ minimizes (12) over $\boldsymbol{\omega}$, its gradient must equals to 0 at $\boldsymbol{\omega}^*$. Thus we have

$$u_{i,l}^* = \exp\big(-M(y_i,l)\mathbf{H}_l^\top(\boldsymbol{x}_i)\boldsymbol{\omega}^*\big) \qquad (15)$$

In our experiments, we have used MOSEK [27] optimization software, which is a primal-dual interior-point solver. Both the primal and dual solutions are available at convergence.

C. Lagrange Dual of AdaBoost.ECC and MultiBoost.ECC

The primal-dual method is so general, actually, arbitrary boosting algorithms based on convex loss functions can be integrated into this framework. Next we investigate another multiclass boosting algorithm: AdaBoost.ECC.

Let us denote $\rho_{i,c}[h^{(t)}] = (M(y_i,t) - M(c,t))\,h^{(t)}(\boldsymbol{x}_i)$. If we define the margin of example $(\boldsymbol{x}_i, y_i)$ on hypothesis $h^{(t)}(\cdot)$ as

$$\begin{aligned}\rho_i[h^{(t)}] &\triangleq \min_{\substack{c\in\mathscr{Y}\\ c\neq y_i}}\{\rho_{i,c}[h^{(t)}]\} \qquad (16) \\ &= M(y_i,t)h^{(t)}(\boldsymbol{x}_i) - \max_{\substack{c\in\mathscr{Y}\\ c\neq y_i}}\{M(c,t)h^{(t)}(\boldsymbol{x}_i)\},\end{aligned}$$

the margin on assembled classifier $\varsigma(\cdot)$ can be computed as

$$\begin{aligned}\rho_i[\varsigma] &\triangleq \min_{\substack{c\in\mathscr{Y}\\ c\neq y_i}}\{\rho_{i,c}[\varsigma]\} \\ &= M(y_i,:)\varsigma(\boldsymbol{x}_i) - \max_{\substack{c\in\mathscr{Y}\\ c\neq y_i}}\{M(c,:)\varsigma(\boldsymbol{x}_i)\} \\ &= \frac{1}{\sum_t \omega^{(t)}}\min_{\substack{c\in\mathscr{Y}\\ c\neq y_i}}\{\sum_{t=1}^{T}\omega^{(t)}\rho_{i,c}[h^{(t)}]\}. \qquad (17)\end{aligned}$$

AdaBoost.ECC tries to maximize the minimum margin by optimizing the following loss function [21]:

$$\begin{aligned} L_{\text{ECC}} &= \sum_{i=1}^{N}\sum_{\substack{c=1\\c\neq y_i}}^{C} \exp\left(-\left(M(y_i,:)-M(c,:)\right)\boldsymbol{f}(\boldsymbol{x}_i)\right) \\ &= \sum_{i=1}^{N}\sum_{\substack{c=1\\c\neq y_i}}^{C} \exp\left(-\sum_{t=1}^{T}\omega^{(t)}\rho_{i,c}[h^{(t)}]\right). \end{aligned} \tag{18}$$

Denote $\mathbf{P}_{i,c} = [\rho_{i,c}[h^{(1)}]\ \rho_{i,c}[h^{(2)}]\cdots\rho_{i,c}[h^{(T)}]]^\top$. Obviously, $\mathbf{P}_{i,y_i} = \mathbf{0}^\top$ for any example $\boldsymbol{x}_i$. Therefore the problem we are interested in can be equivalently written as:

$$\begin{aligned} \min_{\boldsymbol{\omega}} \quad & \sum_{i=1}^{N}\sum_{c=1}^{C}\exp\left(-\mathbf{P}_{i,c}^\top\boldsymbol{\omega}\right) \\ \text{s.t.} \quad & \boldsymbol{\omega}\succeq\mathbf{0}, \|\boldsymbol{\omega}\|_1=\theta. \end{aligned} \tag{19}$$

Like AdaBoost.MO, we have added an ℓ_1 norm constraint to remove the scale ambiguity. Clearly, this is also a convex problem in $\boldsymbol{\omega}$ and strong duality holds.

Theorem 2: The Lagrange dual problem of (19) is

$$\begin{aligned} \max_{r,\boldsymbol{u}} \quad & -r\theta-\sum_{i=1}^{N}\sum_{c=1}^{C}u_{i,c}\log u_{i,c}+\sum_{i=1}^{N}\sum_{c=1}^{C}u_{i,c} \\ \text{s.t.} \quad & \sum_{i=1}^{N}\sum_{c=1}^{C}u_{i,c}\mathbf{P}_{i,c}^\top \preceq r\mathbf{1}^\top, \boldsymbol{u}\succeq\mathbf{0}. \end{aligned} \tag{20}$$

The derivation is very similar with that in the proof of Theorem 1. Notice that the first constraint has no effect on variables u_{i,y_i} since $\mathbf{P}_{i,y_i}=\mathbf{0}^\top$, $\forall i=1,\ldots,N$, however, the problem is still bounded because $-u\log u+u\le 1$ for all $u\ge 0$. Actually, we have $u_{i,y_i}\equiv 1$ in the process of optimization.

To solve this dual problem, we also employ the idea of column generation. Hence at t-th iteration, such an optimal weak classifier can be found by

$$h^*(\cdot)=\operatorname*{argmax}_{h(\cdot)}\sum_{i=1}^{N}\left(\sum_{c}u_{i,c}\left(M(y_i,t)-M(c,t)\right)\right)h(\boldsymbol{x}_i). \tag{21}$$

Notice that $M(y_i,t)-M(c,t) = 2M(y_i,t)\mathbb{1}(M(y_i,t)\neq M(c,t))$, so if we rewrite (21) in the following form:

$$h^*(\cdot)=\operatorname*{argmax}_{h(\cdot)}\sum_{i=1}^{N}\Bigg(\Big(\sum_{c}u_{i,c}\ \mathbb{1}\big(M(y_i,t)\neq M(c,t)\big)\Big) M(y_i,t)h(\boldsymbol{x}_i)\Bigg), \tag{22}$$

it it straightforward to show that we can use the same strategy with AdaBoost.ECC to obtain weak classifiers. To be more precise, the strategy is to minimize the training error with respect to the mislabel distribution.

Looking at optimization problem (6) and (20), they are quite similar. If we denote $\rho_{i,l}[\boldsymbol{h}^{(t)}]=M(y_i,l)h_l^{(t)}(\boldsymbol{x}_i)$ in the first case, the margin of example $(\boldsymbol{x}_i,y_i)$ on hypothesis $\boldsymbol{h}^{(t)}$ would be $\rho_i[\boldsymbol{h}^{(t)}]=\min_l\{\rho_{i,l}[\boldsymbol{h}^{(t)}]\}$, and also

$$\begin{aligned} \rho_i[\varsigma] &= \min_l\{M(y_i,l)f_l(\boldsymbol{x}_i)\} \\ &= \frac{1}{\|\boldsymbol{\omega}\|_1}\min_l\{\sum_t\omega^{(t)}\rho_{i,l}[\boldsymbol{h}^{(t)}]\}. \end{aligned} \tag{23}$$

Based on different definitions of margin, these two problems share exactly the same expression. To summarize, we combine the algorithms that we proposed in this section and give a general framework for multiclass boosting in Algorithm 3.

Algorithm 3 Totally Corrective Multiclass Boosting

```
Input training data $(\boldsymbol{x}_i,y_i)$, $i=1,\ldots,N$; termination
      threshold $\varepsilon>0$; regularization parameter $\theta>0$;
      maximum training iterations $T$.
(1) Initialization
    $t=0$; $\boldsymbol{\omega}=0$; $r=0$;
    $u_{i,k}=\frac{1}{NK}$, $i=1,\ldots,N$, $k=1,\ldots,K$.
while true do
  (2) Find a new weak classifier $\boldsymbol{h}^*(\cdot)$ by solving
      subproblem in column generation:
      $\boldsymbol{h}^*=\operatorname{argmax}_h\sum_i\sum_k u_{i,k}\rho_{i,k}[\boldsymbol{h}]$;
  (3) Check if dual problem is bounded by new constraint:
      if $\sum_i\sum_k u_{i,k}\rho_{i,k}[\boldsymbol{h}^*]<r+\epsilon$, then break;
  (4) Add new constraint to dual problem;
  (5) Solve dual problem to obtain updated $r$ and $\boldsymbol{u}$:
      $\max_{r,\boldsymbol{u}}\ -r\theta-\sum_i\sum_k u_{i,k}\log u_{i,k}+\mathbf{1}^\top\boldsymbol{u}$
        s.t.  $\sum_i\sum_k u_{i,k}\rho_{i,k}[\boldsymbol{h}^{(j)}]\le r$, $j=1,\ldots,t$;
              $\boldsymbol{u}\succeq\mathbf{0}$;
  (6) $t=t+1$; if $t>T$, then break;
end while
(7) Calculate the primal variable $\boldsymbol{\omega}$ according to dual
    solutions and KKT condition.
Output $\boldsymbol{f}(\cdot)=\sum_{j=1}^{t}\omega^{(j)}\boldsymbol{h}^{(j)}(\cdot)$.
```

D. *Hinge Loss Based Multiclass Boosting*

Within this framework, we can devise other boosting algorithms based on different loss functions. Here we give an example. According to (1), if a pattern $\boldsymbol{x}_i$ is well classified, it should satisfy

$$M(y_i,:)\boldsymbol{f}(\boldsymbol{x})\ge M(c,:)\boldsymbol{f}(\boldsymbol{x})+1,\forall c\neq y_i \tag{24}$$

with $\varphi>0$. Define the hinge loss for $\boldsymbol{x}_i$ as

$$\xi_i=\max_c\{M(c,:)\boldsymbol{f}(\boldsymbol{x})+1-\delta_{c,y_i}\}-M(y_i,:)\boldsymbol{f}(\boldsymbol{x}), \tag{25}$$

where $\delta_{c,y_i}=1$ if $c=y_i$, else 0. That is to say, if $\boldsymbol{x}_i$ is fully separable, then $\xi_i=0$; else it suffers a loss $\xi_i>0$. So the problem we are interested in is to find a classifier $\boldsymbol{f}(\cdot)=[\boldsymbol{h}^{(1)}(\cdot)\cdots\boldsymbol{h}^{(T)}(\cdot)]^\top\boldsymbol{\omega}$ through the following optimization:

$$\begin{aligned} \min_{\boldsymbol{\xi},\boldsymbol{\omega}} \quad & \sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & M(c,:)\boldsymbol{f}(\boldsymbol{x})+1-\delta_{c,y_i}-M(y_i,:)\boldsymbol{f}(\boldsymbol{x})\le\xi_i,\forall i,c; \\ & \boldsymbol{\omega}\succeq 0; \|\boldsymbol{\omega}\|_1=\theta. \end{aligned} \tag{26}$$

Theorem 3: The equivalent dual problem of (26) is

$$\min_{r,\boldsymbol{u}} \quad r\theta + \sum_{i=1}^{N}\sum_{c=1}^{C}\delta_{c,y_i}u_{i,c} \tag{27}$$
$$\text{s.t.} \quad \sum_{i=1}^{N}\sum_{c=1}^{C}\boldsymbol{u}_{i,c}\left(M(y_i,:)-M(c,:)\right)\boldsymbol{h}^{(j)}(\boldsymbol{x}_i) \le r, \forall j;$$
$$\boldsymbol{u} \succeq 0; \sum_{c=1}^{C}u_{i,c} = 1, \forall i.$$

Proof: The Lagrangian of this program is a linear function on both $\boldsymbol{\xi}$ and $\boldsymbol{\omega}$, therefore, the proof is easily done by letting the partial derivations on them equal zero, and substituting the results back. ■

Suppose the length of a codeword $M(c,:)$ is L. Using the idea of column generation, we can iteratively obtain weak hypotheses and the associated coefficients by solving

$$\boldsymbol{h}^*(\cdot) = \operatorname*{argmax}_{\boldsymbol{h}(\cdot)} \sum_{i,c}u_{i,c}\sum_{l=1}^{L}\left(M(y_i,t)-M(c,t)\right)\boldsymbol{h}(\boldsymbol{x}_i), \tag{28}$$

or L subproblems

$$h_l^*(\cdot) = \operatorname*{argmax}_{h(\cdot)} \sum_{i=1}^{N}\left(\sum_{c}u_{i,c}\left(M(y_i,l)-M(c,l)\right)\right)h(\boldsymbol{x}_i). \tag{29}$$

This is exactly the same as (21). In other words, we can follow the same procedures as in AdaBoost.ECC to obtain each entry of weak hypotheses.

The proposed boosting framework may inspire us to design other multiclass boosting algorithms in the primal by considering different coding strategies and loss functions. We can use a predefined coding matrix to train a set of multidimensional hypotheses as in AdaBoost.MO, at the same time, penalize the mismatched labels as in AdaBoost.ECC. It seems to be a mixture of these two algorithms. However, this is beyond the scope of this paper.

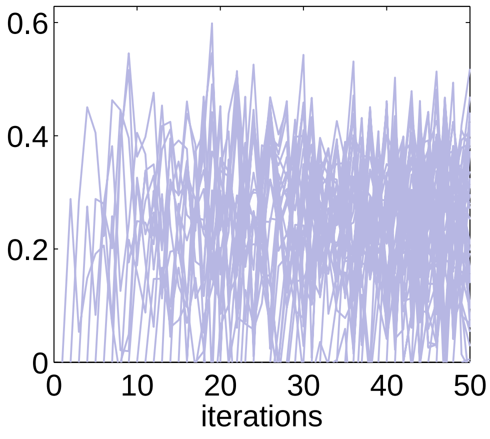
E. Totally Corrective Update

Next, we provide an alternative explanation for the new boosting methods. In Algorithm 3, suppose t weak hypotheses have been found while the cost function still does not converge (*i.e.* , the t-th constraint is not satisfied). To obtain the updated weight distribution $\boldsymbol{u}$ for the next iteration, we need to solve the following optimization problem:

$$\min_{r,\boldsymbol{u}} \quad r\theta + \sum_{i=1}^{N}\sum_{k=1}^{K}u_{i,k}\log u_{i,k} \tag{30}$$
$$\text{s.t.} \quad \boldsymbol{u}^\top\boldsymbol{\rho}[h^{(j)}] \le r, j = 1,\dots,t, \tag{31}$$
$$\boldsymbol{u} \succeq \mathbf{0}, \mathbf{1}^\top\boldsymbol{u} = 1,$$

where $\boldsymbol{u}^\top\boldsymbol{\rho}[h^{(j)}] = \sum_{i=1}^{N}\sum_{k=1}^{K}u_{i,k}\rho_{i,k}[h^{(j)}]$. In other words, ${\boldsymbol{u}^{(t+1)}}^\top\boldsymbol{\rho}[h^{(j)}] \le r$ holds for $j = 1,\dots,t$.

In [22], Kivinen and Warmuth have shown that the *corrective* update of weight distribution in standard boosting algorithms can be viewed as a solution to the divergence minimization problem with the constraint of ${\boldsymbol{u}^{(t+1)}}^\top\boldsymbol{\rho}[h^{(t)}] = 0$.


(a)

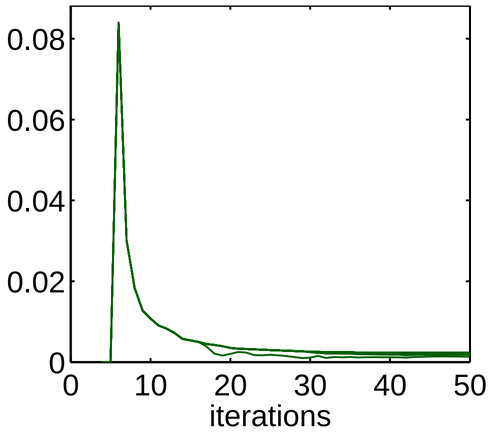

(b)

Fig. 1. Inner products between the new distribution and the past mistake vectors $\mathbf{F}(t) = \{{\boldsymbol{u}^{(t+1)}}^\top\boldsymbol{\rho}[h^{(j)}]\}_{j=1}^{50}$, $t \ge j$ on the training data of *wine*. In (a) AdaBoost.MO, $\mathbf{F}(t)$ starts with $\mathbf{F}(j) = 0$ (corrective update) and quickly becomes uncontrollable; in (b) MultiBoost.MO, $\mathbf{F}(t)$ is consistently bounded by r.

This means that the new distribution $\boldsymbol{u}^{(t+1)}$ should be closest with $\boldsymbol{u}^{(t)}$ but uncorrelated with the *mistakes* made by the current weak hypothesis. If $\boldsymbol{u}^{(t+1)}$ is further required to be orthogonal to all the past t mistake vectors:

$${\boldsymbol{u}^{(t+1)}}^\top\boldsymbol{\rho}[h^{(j)}] = 0, \forall j = 1,\dots,t, \tag{32}$$

the update technique is called a *totally corrective* update. At the same time, the previous t hypotheses receive an increment in their coefficient respectively.

Notice that there is not always an exact solution to (32). Even if a solution exists, this optimization problem might become too complex as t increases. Some attempts have been made to obtain an approximate result. Kivinen and Warmuth [22] suggested using an iterative approach, which is actually a column generation based optimization procedure. Oza [28] proposed to construct $\boldsymbol{u}^{(t+1)}$ by averaging $t+1$ distributions computed from standard AdaBoost update, which is in fact the least-squares solution to linear equations $[\boldsymbol{\rho}[h^{(1)}]\cdots\boldsymbol{\rho}[h^{(t)}]]^\top \boldsymbol{u} = \mathbf{0}$. The stopping criteria of his AveBoost implied a continuous descent of inner products ${\boldsymbol{u}^{(t+1)}}^\top \boldsymbol{\rho}[h^{(j)}]$ as in our algorithm, but exhibited in a heuristic way. Jiang and Ding [29] also noticed the feasibility problem of (32). They proposed to solve a subset, say m equations instead of the entire, however, they did not indicate how to choose m.

In our algorithms, all the coefficients $\{\omega^{(j)}\}_{j=1}^t$ associated to weak hypotheses are updated at each iteration, while the inner products of distribution $\boldsymbol{u}^{(t+1)}$ and $\boldsymbol{\rho}[h^{(j)}]$ are consistently bounded by r. In this sense, our multiclass boosting learning can be considered as a relaxed version of totally corrective algorithm with slack variable r. Figure 1 illustrates the difference between corrective algorithm and our algorithm. Intuitively, it is more feasible to solve inequalities (31) than the same size of equations (32), although most (but not all, see Fig. 1) equalities in (31) are satisfied during the optimization process. Analogous relaxation methods have appeared in LPBoost [11] and TotalBoost [30]. In contrast, our algorithm simultaneously restrains the distribution divergence and correlations of hypotheses. The parameter θ is a trade-off that controls the balance. To our knowledge, this is the first algorithm to introduce the totally corrective concept to multiclass boosting learning.

If we remove the constraints on $\boldsymbol{\omega}$ in primal, for example, constraints (5), the Lagrange dual problem turns to be a totally corrective minimizer of negative entropy:

$$\begin{aligned} \min_{r,\boldsymbol{u}} \quad & \sum_{i=1}^{N}\sum_{k=1}^{K} u_{i,k}\log u_{i,k} \qquad (33)\\ \text{s.t.} \quad & \boldsymbol{u}^\top \boldsymbol{\rho}[h^{(j)}] = 0, j = 1,\ldots,t,\\ & \boldsymbol{u} \succeq \mathbf{0}, \mathbf{1}^\top \boldsymbol{u} = 1, \end{aligned}$$

which is similar to the explanation of corrective update in [22], where the distribution divergence is measured by relative entropy (Kullback-Leibler divergence). However, the constraints on $\boldsymbol{\omega}$ are quite important as we discussed before, and should not be simply removed. Therefore, it seems reasonable to use a relaxed version, instead of standard totally corrective constraints for distribution update of boosting learning.

It has been proven that totally corrective boosting reduces the upper bound on training error more aggressively than standard corrective boosting (apparently including AdaBoost.MO and AdaBoost.ECC) [29], [31], which performs a slowly stage-wise gradient descent procedure on the loss function. Thus, our boosting algorithms can be expected to be faster in convergence than their counterparts. The fast convergence speed is advantageous in reducing the training cost and producing a classifier composed of fewer weak hypotheses [8], [31]. Further, a simplification in strong classifier leads to a speedup of classification, which is critical to many applications, especially those with real-time requirements.

IV. Experiments

In this section, we perform several experiments to compare our totally corrective multiclass boosting algorithms with previous work, including MultiBoost.MO against its stage-wise counterpart AdaBoost.MO, and MultiBoost.ECC against its stage-wise counterpart AdaBoost.ECC. For MultiBoost.MO and AdaBoost.MO, we design error-correcting outputs codes (ECOC) [18] as the coding matrix. For our new algorithms, we solve the dual optimization problems using the off-the-shelf MOSEK package [27].

TABLE I
Multiclass UCI datasets

dataset	#train	#test	#attribute	#class
svmguide2	391	-	20	3
svmguide4	300	312	10	6
wine	178	-	13	3
iris	150	-	4	3
glass	214	-	9	6
thyroid	3772	3428	21	3
dna	2000	1186	180	3
vehicle	846	-	18	4

The datasets used in our experiments are collected from UCI repository [32]. A summary is listed in Table I. We preprocess these datasets as follows: if it is provided with a pre-specified test set, the partitioning setup is retained, otherwise 70% samples are used for training and the other 30% for test. On each test, these two sets are merged and rebuilt by random selecting examples. To keep the balance of multiclass problems, examples associated with the same class are carefully split in proportion. The boosting algorithms are conducted on new sets. This procedure is repeated 20 times. We report the average value as the experimental result.

In the first experiment, we choose decision stumps as the weak learners. As a binary classifier, decision stump is extensively used due to its simplicity. The parameters are preset as follows. The maximum number of training iterations is set to 50, 100 and 500. An important parameter to be tuned is the regularization parameter θ, which equals to the ℓ_1 norm of coefficient vector associated with weak hypotheses. A simple method to choose θ is running the corresponding stage-wise algorithms on the same data and then computing the algebraic sum: $\theta = \sum_j \omega^{(j)}$. For datasets *svmguide2*, *svmguide4*, *wine*, *iris* and *glass*, which contain a small number of examples, we use this method. The same strategy has been used in [8] to test *binary* totally corrective boosting.

For the others, we choose θ from $\{2, 5, 8, 10, 12, 15, 20, 30, 40, 45, 60, 80, 100, 120, 150, 200\}$ by running a five-fold cross validation on training data. In particular, we use a pseudo-random code generator in the cross validations of AdaBoost.ECC and MultiBoost.ECC, to make sure each candidate parameter is tested under the same coding strategy.

The experimental results are reported in Table II. As we can see, almost all the training errors of totally corrective algorithms are lower than their counterparts, except in the case both algorithms have converged to 0. In Fig. 2, we show the training error curves of some datasets when the training iteration number is set to 500. Obviously, the convergence speed of our totally corrective boosting is much faster than the stage-wise one. This conclusion is consistent with the discussion in Section III-E. Especially on *svmguide2*, *iris* and *glass*, new algorithms are around 50 iterations faster than their

TABLE II
TRAINING AND TEST ERRORS(INCLUDING MEAN AND STANDARD DEVIATION) OF ADABOOST.MO, MULTIBOOST.MO, ADABOOST.ECC AND MULTIBOOST.ECC ON UCI DATASETS. AVERAGE RESULTS OF 20 REPEATED TESTS ARE REPORTED. BASE LEARNERS ARE DECISION STUMPS. RESULTS IN BOLD ARE BETTER THAN THEIR COUNTERPARTS.

dataset	algorithm	train error 50	train error 100	train error 500	test error 50	test error 100	test error 500
svmguide2	AB.MO	0.016±0.005	0.000±0.000	0±0	0.225±0.031	**0.212±0.030**	**0.229±0.024**
	TC.MO	**0.011±0.003**	**0±0**	0±0	**0.224±0.024**	0.221±0.038	0.233±0.028
	AB.ECC	0.049±0.009	0.003±0.003	0±0	0.253±0.032	0.226±0.031	0.226±0.030
	TC.ECC	**0.032±0.009**	**0.001±0.002**	0±0	**0.242±0.031**	**0.223±0.034**	**0.221±0.028**
svmguide4	AB.MO	0.041±0.004	0.016±0.002	0±0	0.204±0.025	**0.194±0.017**	**0.190±0.017**
	TC.MO	**0.039±0.004**	**0.014±0.002**	0±0	**0.203±0.023**	0.196±0.016	0.193±0.018
	AB.ECC	0.180±0.021	0.115±0.023	0±0	0.285±0.023	0.262±0.025	**0.233±0.023**
	TC.ECC	**0.158±0.023**	**0.087±0.020**	0±0	**0.275±0.022**	**0.245±0.028**	0.237±0.022
wine	AB.MO	0±0	0±0	0±0	0.032±0.015	**0.030±0.029**	**0.031±0.028**
	TC.MO	0±0	0±0	0±0	0.032±0.016	0.031±0.023	0.032±0.026
	AB.ECC	0±0	0±0	0±0	0.026±0.020	0.032±0.015	0.026±0.024
	TC.ECC	0±0	0±0	0±0	0.026±0.022	0.032±0.034	0.026±0.019
iris	AB.MO	0.000±0.001	0±0	0±0	0.062±0.026	**0.064±0.025**	0.060±0.032
	TC.MO	**0±0**	0±0	0±0	**0.062±0.026**	0.067±0.023	**0.057±0.025**
	AB.ECC	0±0	0±0	0±0	**0.057±0.024**	**0.062±0.026**	0.053±0.032
	TC.ECC	0±0	0±0	0±0	0.061±0.021	0.067±0.020	**0.051±0.028**
glass	AB.MO	0.026±0.003	0.003±0.001	0±0	**0.275±0.034**	**0.246±0.061**	**0.268±0.051**
	TC.MO	**0.022±0.003**	**0.002±0.001**	0±0	0.280±0.039	0.252±0.059	0.273±0.047
	AB.ECC	0.168±0.032	0.078±0.018	0±0	0.352±0.052	0.313±0.043	**0.298±0.044**
	TC.ECC	**0.113±0.030**	**0.020±0.013**	0±0	**0.327±0.048**	**0.302±0.035**	0.306±0.045
thyroid	AB.MO	0.003±0.001	0.001±0.000	0±0	0.006±0.001	**0.006±0.001**	**0.006±0.002**
	TC.MO	**0.001±0.001**	**0.001±0.001**	0±0	**0.006±0.001**	0.006±0.001	0.006±0.001
	AB.ECC	0.006±0.001	0.002±0.001	0±0	0.010±0.002	0.008±0.002	0.005±0.001
	TC.ECC	**0.000±0.001**	**0±0**	0±0	**0.006±0.002**	**0.006±0.002**	**0.004±0.000**
dna	AB.MO	0.053±0.002	0.040±0.002	**0.028±0.002**	**0.076±0.007**	0.066±0.006	0.061±0.006
	TC.MO	**0.052±0.004**	**0.039±0.003**	0.029±0.002	0.078±0.007	**0.064±0.006**	**0.054±0.005**
	AB.ECC	0.070±0.004	0.049±0.005	**0.017±0.004**	0.089±0.008	0.077±0.009	0.069±0.005
	TC.ECC	**0.059±0.005**	**0.041±0.006**	0.028±0.005	**0.083±0.008**	**0.070±0.006**	**0.065±0.004**
vehicle	AB.MO	0.099±0.003	0.073±0.003	**0.018±0.000**	0.249±0.020	0.245±0.019	0.212±0.010
	TC.MO	**0.086±0.009**	**0.048±0.007**	0.019±0.027	**0.241±0.020**	**0.231±0.018**	**0.211±0.021**
	AB.ECC	0.271±0.010	0.207±0.011	0.096±0.010	0.359±0.022	0.300±0.021	**0.249±0.017**
	TC.ECC	**0.208±0.018**	**0.140±0.019**	**0.055±0.011**	**0.327±0.022**	**0.287±0.024**	0.257±0.028

counterparts.

In terms of test error, it is not apparent which algorithm is better. Empirically speaking, the totally corrective boosting has a comparable generalization capability with the stage-wise version. It is noticeable that on *thyroid*, *dna* and *vehicle*, where the regularization parameter is adjusted via cross-validation, our algorithms clearly outperform their counterparts. We conjecture that if we tune this parameter more carefully, the performance of new algorithms could be further improved.

In the second experiment, we change the base learner with another binary classifier: Fisher's linear discriminant function (LDA). For simplicity, we only run AdaBoost.ECC and MultiBoost.ECC at this time. All the parameters and settings are the same as in the first experiment. The results are reported in Table III. Again, the convergence speed of totally corrective boosting is faster than gradient descent version. We also notice that two algorithms of ECCs are better with LDAs than with decision stumps on *svmguide2*, but worse on *svmguide4*, *glass*, *thyroid* and *vehicle*, although LDA is evidently stronger than decision stump.

To further verify the generalization capability of our multiclass boosting algorithms, we run the Wilcoxon rank-sum test [33] on test errors in Tables II and III. Wilcoxon rank-sum test is a nonparametric statistical tool for assessing the hypothesis that two sets of samples are drawn from the identical distribution. If the totally corrective boosting is comparable with its counterpart in classification error, the Wilcoxon test is supposed to output a higher significant probability. The results are reported in Table IV. We can see the probabilities are higher enough (> 0.8) to claim the identity, except in the case ECC algorithms with decision stumps when $T = 50$ and ECC algorithms with LDAs when $T = 500$. However, if we take a close look at those two cases, we can find where our totally corrective algorithms perform better than their counterparts.

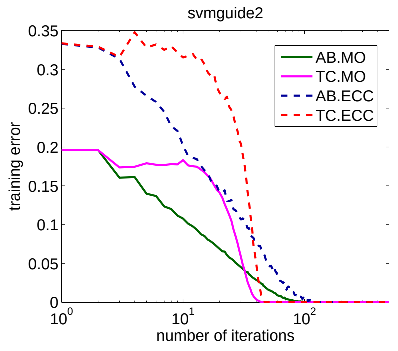
TABLE IV
WILCONXON RANK-SUM TEST ON CLASSIFICATION ERRORS

algorithms	$T = 50$	$T = 100$	$T = 500$
MOs with stumps	0.902	0.878	1.000
ECCs with stumps	0.743	0.821	0.983
ECCs with LDAs	0.959	1.000	0.798

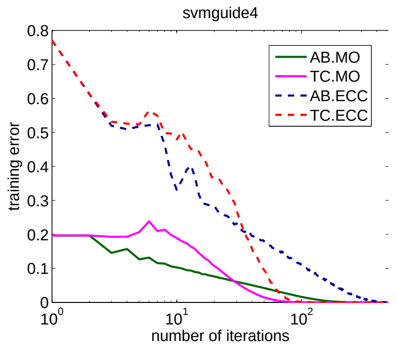
Next, we investigate the minimum margin of training examples, which has a close relationship with the generalization error [10]. Warmuth and Rätsch [30] have proven that by introducing a slack variable r, totally corrective boosting can realize a larger margin than corrective version with the same number of weak hypotheses. We test this conclusion on datasets *svmguide2*, *svmguide4* and *iris*. At each iteration, we record the minimum margin of training examples on the current combination of weak hypotheses. The results are illustrated in Fig. 3. It should be noted in algorithms of MOs and ECCs, the definitions of margin are different: (23) and (17) respectively. However, it is clear that in any case, totally corrective boosting algorithms increase the margin much faster

TABLE III
TRAINING AND TEST ERRORS (INCLUDING MEAN AND STANDARD DEVIATION) OF ADABOOST.ECC AND MULTIBOOST.ECC. THE AVERAGE RESULTS OF 20 REPEATED TESTS ARE REPORTED. BASE LEARNERS ARE FISHER'S LINEAR DISCRIMINANT FUNCTIONS.

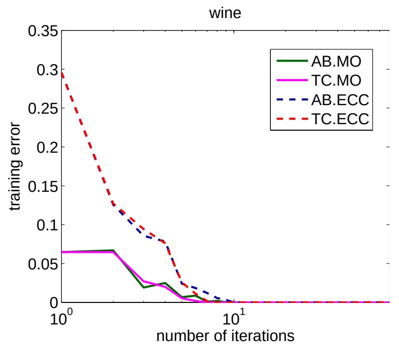
dataset	algorithm	train error 50	train error 100	train error 500	test error 50	test error 100	test error 500
svmguide2	AB.ECC	0.055±0.021	0.003±0.006	0±0	**0.214±0.033**	**0.224±0.025**	**0.197±0.021**
	TC.ECC	**0.031±0.013**	**0±0**	0±0	0.228±0.044	0.229±0.024	0.221±0.027
svmguide4	AB.ECC	0.323±0.038	0.208±0.045	0.000±0.001	0.457±0.036	0.428±0.049	**0.328±0.037**
	TC.ECC	**0.275±0.035**	**0.149±0.033**	**0±0**	**0.418±0.041**	**0.377±0.051**	0.338±0.048
wine	AB.ECC	0±0	0±0	0±0	0.038±0.025	0.027±0.025	0.030±0.021
	TC.ECC	0±0	0±0	0±0	**0.025±0.027**	**0.020±0.022**	**0.015±0.014**
iris	AB.ECC	0±0	0±0	0±0	**0.041±0.030**	**0.040±0.024**	0.056±0.038
	TC.ECC	0±0	0±0	0±0	0.046±0.037	0.042±0.026	**0.049±0.038**
glass	AB.ECC	0.194±0.033	0.080±0.021	0±0	0.384±0.051	**0.357±0.051**	0.369±0.052
	TC.ECC	**0.077±0.028**	**0.001±0.002**	0±0	**0.374±0.046**	0.364±0.047	**0.359±0.058**
thyroid	AB.ECC	0.041±0.004	0.035±0.005	0.001±0.001	0.048±0.006	0.046±0.004	0.032±0.004
	TC.ECC	**0.033±0.006**	**0.018±0.010**	**0±0**	**0.043±0.007**	**0.040±0.004**	**0.030±0.001**
dna	AB.ECC	0.000±0.000	0.000±0.000	0.000±0.000	0.081±0.007	0.084±0.009	0.077±0.010
	TC.ECC	0.000±0.000	0.000±0.000	**0.000±0.000**	**0.068±0.008**	**0.065±0.007**	**0.064±0.008**
vehicle	AB.ECC	0.237±0.012	0.176±0.018	0.004±0.002	**0.301±0.017**	**0.297±0.021**	0.276±0.026
	TC.ECC	**0.196±0.015**	**0.125±0.011**	**0±0**	0.312±0.029	0.313±0.027	**0.272±0.037**



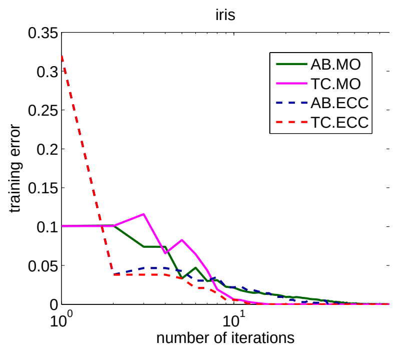
(a)



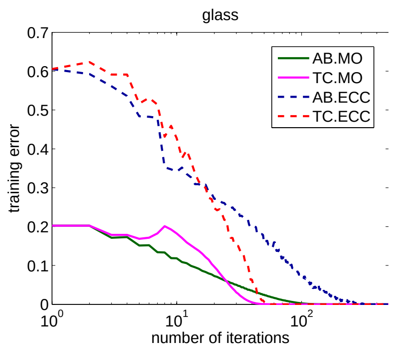
(b)



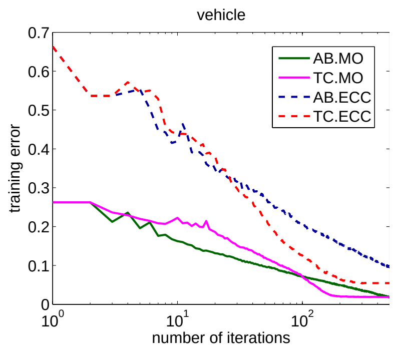
(c)



(d)



(e)



(f)

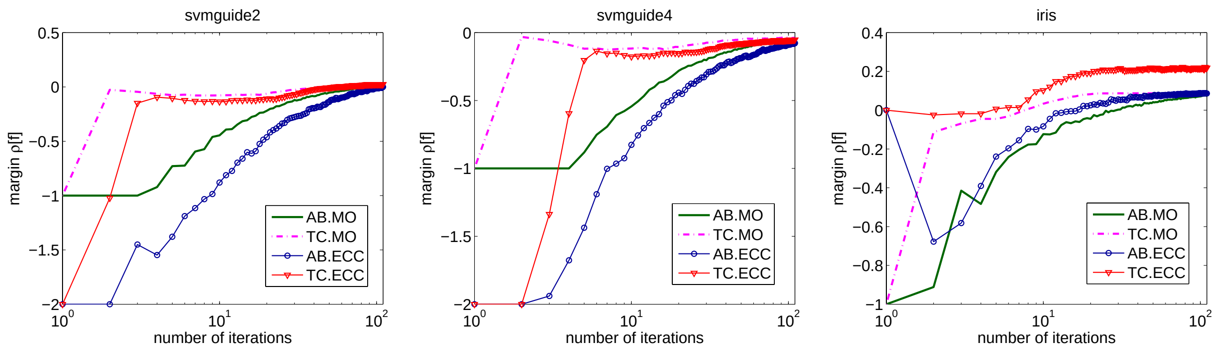
Fig. 2. Training error curves of AdaBoost.MO, MultiBoost.MO, AdaBoost.ECC and MultiBoost.ECC on *svmguide2*, *svmguide4*, *wine*, *iris*, *glass* and *vehicle*. The number of training iterations is 500. Base learners are decision stumps.

than the two previous ones.

V. DISCUSSION AND CONCLUSION

We have presented two boosting algorithms for multiclass learning, which are mainly based on derivations of the Lagrange dual problems for AdaBoost.MO and AdaBoost.ECC. Using the column generation technique, we design new totally corrective boosting algorithms. The two algorithms can be formulated into a general framework base on the concept of margins. Actually, this framework can also incorporate other multiclass boosting algorithms, such as SAMME. In this paper, however, we have focused on multiclass boosting with binary weak learners.

Furthermore, we indicate that the proposed boosting algorithms are totally corrective. This is the first time to use this concept in multiclass boosting learning. We also discuss the



(a) (b) (c)

Fig. 3. The minimum margin on assembled classifiers of AdaBoost.MO, MultiBoost.MO, AdaBoost.ECC and MultiBoost.ECC. The definitions of margin are different in MOs and ECCs, however, it clearly shows that the totally corrective algorithms realize a larger margin than their counterparts within the same iterations.

reason of introducing slack variables. Experiments on UCI datasets show that our new algorithms are much faster than their gradient descent counterparts in terms of convergence speed, but comparable with them in classification capability. The experimental results also demonstrate that totally corrective algorithms can maximize the example margin more aggressively.

References

[1] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comp. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[2] E. Ong and R. Bowden, "A boosted classifier tree for hand shape detection," in *Proc. IEEE Int. Conf. Automatic Face & Gesture Recogn.*, 2004, pp. 889–894.

[3] K. Okuma, A. Taleghani, N. Freitas, J. Little, and D. Lowe, "A boosted particle filter: Multitarget detection and tracking," *Proc. Eur. Conf. Comp. Vis.*, pp. 28–39, 2004.

[4] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer, "Generic object recognition with boosting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 416–431, 2006.

[5] P. Dollár, Z. Tu, H. Tao, and S. Belongie, "Feature mining for image classification," in *IEEE Conf. Comp. Vis. Pattern Recogn.*, 2007.

[6] K. Tieu and P. Viola, "Boosting image retrieval," *Int. J. Comp. Vis.*, vol. 56, no. 1, pp. 17–36, 2004.

[7] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, no. 3, pp. 297–336, 1999.

[8] C. Shen and H. Li, "On the dual formulation of boosting algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.47

[9] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–374, 2000.

[10] R. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Ann. Statist.*, vol. 26, no. 5, pp. 1651–1686, 1998.

[11] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Mach. Learn.*, vol. 46, no. 1, pp. 225–254, 2002.

[12] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Mach. Learn.*, 1996, pp. 148–156.

[13] J. Zhu, S. Rosset, H. Zou, and T. Hastie, "Multi-class AdaBoost," *Ann Arbor*, vol. 1001, p. 48109, 2006.

[14] J. Huang, S. Ertekin, Y. Song, H. Zha, and C. Giles, "Efficient multiclass boosting classification with active learning," in *Proc. SIAM Int. Conf. Data Min.*, 2007, pp. 297–308.

[15] M. Skurichina and R. Duin, "Bagging, boosting and the random subspace method for linear classifiers," *Pattern Anal. Appl.*, vol. 5, no. 2, pp. 121–135, 2002.

[16] D. Masip and J. Vitria, "Boosted linear projections for discriminant analysis," *Frontiers in Artificial Intelligence and Applications*, pp. 45–52, 2005.

[17] B. Kégl and R. Busa-Fekete, "Boosting products of base classifiers," in *Proc. Int. Conf. Mach. Learn.* ACM, 2009, pp. 497–504.

[18] T. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *J. Artif. Intell. Res.*, vol. 2, pp. 263–286, 1995.

[19] R. E. Schapire, "Using output codes to boost multiclass learning problems," in *Proc. Int. Conf. Mach. Learn.*, 1997, pp. 313–321.

[20] V. Guruswami and A. Sahai, "Multiclass learning, boosting, and error-correcting codes," in *Proc. Annual Conf. Learn. Theory*, 1999, pp. 145–155.

[21] Y. Sun, S. Todorovic, and J. Li, "Unifying multi-class adaboost algorithms with binary base learners under the margin framework," *Pattern Recogn. Lett.*, vol. 28, no. 5, pp. 631–643, 2007.

[22] J. Kivinen and M. K. Warmuth, "Boosting as entropy projection," in *Proc. Annual Conf. Learn. Theory*, 1999, pp. 134–144.

[23] K. Crammer and Y. Singer, "On the algorithmic implementation of multi-class SVMs," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, 2001.

[24] E. Allwein, R. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *J. Mach. Learn. Res.*, vol. 1, pp. 113–141, 2001.

[25] K. Crammer and Y. Singer, "On the learnability and design of output codes for multiclass problems," *Mach. Learn.*, vol. 47, no. 2, pp. 201–233, 2002.

[26] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ Press, 2004.

[27] "The mosek optimization software," 2010. [Online]. Available: http://www.mosek.com

[28] N. Oza, "Boosting with averaged weight vectors," in *Proc. Int. Conf. Mult. Classifier*. Springer-Verlag, 2003, pp. 15–24.

[29] Y. Jiang and X. Ding, "Partially corrective adaboost," *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 469–478, 2010.

[30] M. Warmuth, J. Liao, and G. Rätsch, "Totally corrective boosting algorithms that maximize the margin," in *Proc. Int. Conf. Mach. Learn.*, 2006, p. 1008.

[31] J. Sochman and J. Malas, "Adaboost with totally corrective updates for fast face detection," in *Proc. IEEE Int. Conf. Automatic Face & Gesture Recogn.*, 2004, pp. 445–450.

[32] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998." [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[33] F. Wilcoxon, S. K. Katti, and R. A. Wilcox, "Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test," *Selected tables in mathematical statistics*, vol. 1, pp. 171–259, 1970.

PLACE
PHOTO
HERE

Zhihui Hao is a Ph.D. student in the School of Automation at Beijing Institute of Technology, China and currently visiting the Australian National University and NICTA, Canberra Research Laboratory, Australia. He received the B.E. degree in automation at Beijing Institute of Technology in 2006. His research interests include object detection, tracking, and machine learning in computer vision.

PLACE
PHOTO
HERE

Chunhua Shen completed the Ph.D. degree from School of Computer Science, University of Adelaide, Australia in 2005; and the M.Phil. degree from Mathematical Sciences Institute, Australian National University, Australia in 2009. Since Oct. 2005, he has been working with the computer vision program, NICTA (National ICT Australia), Canberra Research Laboratory, where he is a senior research fellow and holds a continuing research position. His main research interests include statistical machine learning and its applications in computer vision and image processing.

PLACE
PHOTO
HERE

Nick Barnes completed a Ph.D. in 1999 at the University of Melbourne. In 1999 he was a visiting researcher at the LIRA-Lab, Genoa, Italy. From 2000 to 2003, he lectured in Computer Science and Software Engineering, at the University of Melbourne. He is now a principal researcher at NICTA Canberra Research Laboratory in computer vision.

PLACE
PHOTO
HERE

Bo Wang received the Ph.D. degree from the Department of Control Theory and Engineering at Beijing Institute of Technology (BIT). He is currently working in the School of Automation at BIT. His main research interests include information detection and state control for high-speed moving objects.