Appearing in IEEE Conf. Comp. Vis. Pattern Recogn. 2011. This reprint differs from the original in pagination and typographic detail.

A Direct Formulation for Totally-corrective Multi-class Boosting

Chunhua Shen[1,2*] Zhihui Hao[3†]
[1] Australian Center for Visual Technologies, University of Adelaide, SA 5005, Australia
[2] NICTA[‡], Canberra Research Laboratory, ACT 2600, Australia
[3] Beijing Institute of Technology, Beijing 100081, China

Abstract

Boosting combines a set of moderately accurate weak classifiers to form a highly accurate predictor. Compared with binary boosting classification, multi-class boosting received less attention. We propose a novel multi-class boosting formulation here. Unlike most previous multi-class boosting algorithms which decompose a multi-boost problem into multiple independent binary boosting problems, we formulate a direct optimization method for training multi-class boosting. Moreover, by explicitly deriving the Lagrange dual of the formulated primal optimization problem, we design totally-corrective boosting using the column generation technique in convex optimization. At each iteration, all weak classifiers' weights are updated. Our experiments on various data sets demonstrate that our direct multi-class boosting achieves competitive test accuracy compared with state-of-the-art multi-class boosting in the literature.

1. Introduction

Boosting has attracted much research interest recently in computer vision due to its successful applications, among which real-time object detection is a typical example [18]. To explain why boosting works, Schapire *et al.* [13] introduced the margin theory, which is inspired by the margin theory in support vector machines, and suggested that boosting is especially effective at maximizing the minimum margin of training data. Based on this idea, LPBoost [5] is designed to maximize the relaxed minimum margin (soft margin) using the hinge loss.

Multi-class boosting is important in the sense that most classification tasks in the real world have multiple classes. Multi-class boosting is less understood, compared to its binary counterpart. So far, most boosting algorithms are designed for binary classification. The most natural strategy for building multi-class boosting is to partition a multi-class problem into a set of independent binary classification problems. Each binary classifier's responsibility is to distinguish one of the class labels against the others. Output codes based methods belong to this category. Albeit the output codes provide a simple and intuitive solution to multi-class classification, they completely ignore the pairwise correlation information between different classes.

In this work, we proffer a direct approach for learning multi-class boosting. We generalize the concept of separation hyperplane and margin in boosting for multi-class problems. This is the basis of our new multi-class boosting. Similar ideas have been used in multi-class support vector machines [3, 7, 19]. To our knowledge, it has not been employed to design *totally-corrective* multi-class boosting.

The key idea of our approach is that, given an example $\{\boldsymbol{x}, y\}$, the decision function with the correct label $F_y(\boldsymbol{x})$ must be larger than the decision function's value with an incorrect label $F_r(\boldsymbol{x}), \forall r \neq y$. We then formulate a convex optimization problem, which tries to maximize $F_y(\boldsymbol{x}) - F_r(\boldsymbol{x})$ as much as possible in a regularized framework. This leads to a constrained semi-infinite convex optimization problem, which may have infinitely many variables. In order to design a boosting algorithm, we explicitly derive its Lagrange dual and column generation is then used to solve the optimization problem iteratively. When the hinge loss is used, our formulation can be viewed as a direct extension of LPBoost [5] to the multi-class case. We also discuss the case using the exponential loss. In theory, any other convex loss function can be employed here, same as in the binary classification case. Note that the resulting optimization problems do not fit into the AnyBoost framework [11], for which it is not clear how to cope with multiple constraints. In short, our main contributions are as follows. We propose a novel direct approach to multi-class boosting formulation based on the generalization of

*C. Shen's participation in this research was supported by the Australian Research Council through its Special Research Initiative in Bionic Vision Science and Technology grant to Bionic Vision Australia.

†Z. Hao's contribution was made when visiting NICTA, Canberra Research Laboratory.

‡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.

the conventional "margin" in binary classification. The proposed boosting is totally corrective in the sense that all the coefficients of learned weak classifiers are updated at each iteration. Since multi-class classification can be seen as an instance of structured learning problems [17], the proposed formulation may also be applicable to other structured prediction problems.

Related work We briefly review most relevant work on multi-class boosting before we present our algorithms. AdaBoost is the first practical *binary* boosting algorithm proposed by Freund and Schapire [8]. A requirement on the weak classifiers for binary AdaBoost is that a weak classifier's accuracy should be higher than 0.5. That is to say that a weak classifier must be better than random guessing in terms of classification capability. AdaBoost.M1, a direct extension of AdaBoost to multi-class by requiring the weak classifiers to be capable of classifying multi-class problems. However, the requirement that a weak classifier's weighted error must be better than 0.5 is still needed, which can be hard to achieve for problems with many classes. For a problem with k classes, random guessing can only guarantee an accuracy of $1/k$. To alleviate this difficulty, a solution is to decompose a multi-class boosting problem into a few binary classification problems. To this end, strategies like "one-against-all" and "one-against-one" can be employed. These two approaches can be viewed as special cases of error-correcting output coding (ECOC) [4, 6]. By introducing a coding matrix, AdaBoost.MO [14] is a typical example of ECOC based multi-class boosting. The learned classifier is multi-dimensional, with each entry boosted on a relabeled set of training data. So, algorithms in this category include AdaBoost.MO [14], AdaBoost.OC and AdaBoost.ECC [9]. AdaBoost.OC can be seen as a variant of AdaBoost.MO which also combines boosting and ECOC. However, unlike AdaBoost.MO, AdaBoost.OC uses a collection of randomly generated codewords. For more details about the random codes, see [15] for details.

Transforming the multi-class classification problem into a bunch of binary classification problems is easier to implement in that the weak classifiers are binary classifiers.

The SAMME algorithm of Zhu *et al*. [20] is an extension of AdaBoost. Similar to ours, SAMME does not reduce multi-class to multiple binary problems but instead optimizes a multi-class exponential loss function. It requires the weak classifiers to achieve less error than uniform random guessing for multiple labels ($1/k$ for k labels). When $k = 2$, SAMME reduces to the standard AdaBoost. A condition for SAMME is that only multi-class weak classifiers like decision trees can be used. In contrast, our proposed algorithms use binary weak classifiers. By multi-class weak classifiers, we mean that the employed weak hypotheses should be able to give predictions on all k possible labels at each call. Usually they are complicated and time-consuming for training compared with simple binary learners. A higher complexity of assembled classifier also often implies a larger risk of over-fitting the training data.

Our work here can also be seen as an extension of the general totally-corrective boosting framework of [16] to the multi-class case.

Notation A bold lowercase letter ($\boldsymbol{u}$, $\boldsymbol{v}$) denotes a column vector. An uppercase letter (U, V) denotes a matrix. $\mathbf{Tr}(U)$ is the trace of a symmetric matrix. An element-wise inequality between two vectors or matrices like $\boldsymbol{u} \geq \boldsymbol{v}$ means $u_i \geq v_i$ for all i.

Let $(\boldsymbol{x}_i; y_i) \in \mathbb{R}^d \times \{1, 2, \cdots, k\}, i = 1 \cdots m$, be a set of m multi-class training examples. We have k classes here. We denote $\mathcal{H}$ a set of weak classifiers (dictionary); the size of $\mathcal{H}$ can be infinite. Each $h_j(\cdot) \in \mathcal{H}, j = 1 \ldots n$, is a function that maps an input $\boldsymbol{x}$ to $\{-1, +1\}$. Although our discussion works for the general case that $h(\cdot)$ can be any real value, we use binary weak classifiers in this work. The matrix $H \in \mathbb{R}^{m \times n}$ denotes the weak classifiers' response on the whole training data; i.e., its (i, j) entry is $H_{ij} = h_j(\boldsymbol{x}_i)$. Therefore each column $H_{:j}$ contains the output of weak classifier $h_j(\cdot)$ on the entire training set and each row $H_{i:}$ is the outputs of all weak classifiers on the ith training datum $\boldsymbol{x}_i$. $\|W\|_1 = \sum_{ij} |W_{ij}|$ is the ℓ_1 norm.

Boosting algorithms learn a strong classifier of the form $F(\boldsymbol{x}) = \sum_{j=1}^{n} w_j h_j(\boldsymbol{x})$ which is parameterized by a vector $\boldsymbol{w} \in \mathbb{R}^n$. In the multi-class setting, we need to learn a classifier for each class. So for class r, $(r = 1, \cdots, k)$, the learned strong classifier is $F_r(\boldsymbol{x}) = \sum_{j=1}^{n} w_{r,j} h_j(\boldsymbol{x})$ with the parameter $\boldsymbol{w}_r$. We define $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_k] \in \mathbb{R}^{n \times k}$. Here we assume that the weak classifier dictionary for each class is the same.

The remaining content is structured as follows. Section 2 presents the main algorithm of our work. In particular, we start from deriving our algorithm from the piece-wise linear hinge loss. Then we discuss the exponential loss case as well and we generalize the proposed method to any convex loss. We present our experimental results in Section 3 and conclude our work in Section 4.

2. A direct formulation for multi-class boosting

In binary classification, the margin is defined as $yF(\boldsymbol{x})$ with $y \in \{-1, +1\}$. In the framework of maximum margin learning, one tries to maximize the margin $yF(\boldsymbol{x})$ as much as possible. A large margin implies the learned classifier confidently classifies the corresponding training example. We generalize this idea to multi-class problems in this section.

2.1. The hinge loss

Let us consider the hinge loss case, which is piecewise linear and therefore makes it easy to derive our formulation. As we will show, both the primal and dual problems

are linear programs (LPs), which can be globally solved in polynomial time. The basic idea is to learn classifiers by pairwise comparison. For a training example $(\boldsymbol{x}, y)$, if we have a perfect classification rule, then the following holds

$$F_y(\boldsymbol{x}) > F_r(\boldsymbol{x}), \text{ for any } r \neq y.$$

In the large margin framework with the hinge loss, ideally

$$F_y(\boldsymbol{x}) \geq 1 + F_r(\boldsymbol{x}), \text{ for any } r \neq y, \tag{1}$$

should be satisfied. This means that the correct label is supposed to have a classification confidence that is larger by at least a unit than any of the confidences for the other predictions. This extension of "margin" to the multi-class case has been introduced in support vector machines [7, 19]. As pointed out in [19], to formulate multi-class problems as a pairwise ranking problem in a single optimization can be more powerful than to solve a bunch of one-versus-rest binary classifications. The argument is that we may generate a multi-class dataset that can be classified perfectly by the decision rule of type (9), but for which the training data cannot be separated with no error by one-versus-rest.

By introducing the indication operator $\delta_{s,t}$ such that $\delta_{s,t} = 1$ if $s = t$ and $\delta_{s,t} = 0$ otherwise, the above equation can be simplified as

$$\delta_{r,y} + F_y(\boldsymbol{x}) \geq 1 + F_r(\boldsymbol{x}), \forall r = 1, 2, \cdots, k. \tag{2}$$

We generalize this idea to the entire training set and introduce slack variables $\boldsymbol{\xi}$ to enable soft-margin. The primal problem that we want to optimize can then be written as

$$\begin{aligned} \min_{W,\boldsymbol{\xi}} \; & \sum_{i=1}^{m} \xi_i + \nu \left\| W \right\|_1 \\ \text{s.t. } & \delta_{r,y_i} + H_{i:}\boldsymbol{w}_{y_i} \geq 1 + H_{i:}\boldsymbol{w}_r - \xi_i, \forall i, r, \\ & W \geq 0. \end{aligned} \tag{3}$$

Here $\nu > 0$ is the regularization parameter. $\boldsymbol{\xi} \geq 0$ always holds. If for a particular $\boldsymbol{x}_i$, ξ_i is negative, then one of the constraint in (3) that corresponds to the case $r = y_i$ will be violated. In other words, the constraint corresponding to the case $r = y_i$ ensures the non-negativeness of $\boldsymbol{\xi}$.

Note that we have one slack variable for each training example. It is also possible to assign a slack variable to each constraint in (3).

We derive its Lagrange dual, similar to case of LPBoost [5]. The Lagrangian of problem (3) can be written as

$$\begin{aligned} L = & \sum_i \xi_i + \nu \sum_{j,r} W_{jr} - \sum_{i,r} U_{ir} \cdot \\ & \left(\delta_{r,y_i} + H_{i:}\boldsymbol{w}_{y_i} - 1 - H_{i:}\boldsymbol{w}_r + \xi_i\right) - \mathbf{Tr}(V^\top W), \end{aligned}$$

with $U \geq 0$, $V \geq 0$. At optimum, the first derivative of the Lagrangian w.r.t. the primal variables must vanish,

$$\frac{\partial L}{\partial \xi_i} = 0 \longrightarrow \sum_r U_{ir} = 1, \forall i. \tag{4}$$

Also,

$$\frac{\partial L}{\partial \boldsymbol{w}_r} = \mathbf{0} \tag{5}$$

$$\longrightarrow \nu 1^\top + \sum_r U_{ir} H_{i:} - \sum_{i,r=y_i} \overbrace{\left(\sum_l U_{il}\right)}^{=1, \text{ due to (4)}} H_{i:} = V_{r:},$$

which leads to $\sum_i U_{ir} H_{i:} - \sum_i \delta_{r,y_i} H_{i:} \geq -\nu \mathbf{1}^\top, \forall r$. So the Lagrange dual can be written as:[1]

$$\begin{aligned} \min_U \; & \sum_{r=1}^{k} \sum_{i=1}^{m} \delta_{r,y_i} U_{ir} \\ \text{s.t. } & \sum_i \left(\delta_{r,y_i} - U_{ir}\right) H_{i:} \leq \nu \mathbf{1}^\top, \forall r, \\ & \sum_r U_{ir} = 1, \forall i; \; U \geq 0. \end{aligned} \tag{6}$$

Each row of the matrix U is normalized. The first set of constraints can be infinitely many:

$$\sum_i \left(\delta_{r,y_i} - U_{ir}\right) h(\boldsymbol{x}_i) \leq \nu, \forall r, \text{ and } \forall h(\cdot) \in \mathcal{H}. \tag{7}$$

We can now use column generation to solve the problem, similar to the LPBoost [5]. The subproblem for generating weak classifiers is

$$h^*(\cdot) = \operatorname*{argmax}_{h(\cdot)} \sum_{i=1}^{m} \left(\delta_{r,y_i} - U_{ir}\right) h(\boldsymbol{x}_i). \tag{8}$$

The matrix $U \in \mathbb{R}^{m \times k}$ plays the role of measuring importance of a training example.

The following algorithm can be used to implement our hinge loss based MULTIBOOST.

Algorithm 1: MULTIBOOST **with the hinge loss**

Initialize each entry of U to be $1/k$.

loop

– Find the weak classifier by solving the subproblem (8), and add this weak classifier to the primal problem.

– Solve the primal problem (3) using a primal-dual interior-point LP solver such as Mosek [1], such that the dual solution is also available.

until convergence

[1] Strictly speaking, this is one of the Lagrange duals of the original primal because some transformations from the standard form have been performed.

The output is the learned multi-class strong classifier. The classification rule is

$$r^* = \operatorname*{argmax}_{r=1}^{k} \sum_{j=1}^{n} w_{r,j} h_j(\boldsymbol{x}), \tag{9}$$

for a test instance $\boldsymbol{x}$.

2.2. The exponential loss

Now let us consider the exponential loss in the section. In the case of the exponential loss, We may write the primal optimization problem as

$$\begin{aligned} &\min_{W} \sum_{i=1}^{m}\sum_{r=1}^{k} \exp\left[-\left(H_{i:}\boldsymbol{w}_{y_i} - H_{i:}\boldsymbol{w}_r\right)\right] + \nu' \left\|W\right\|_1 \\ &\text{s.t. } W \geq 0. \end{aligned} \tag{10}$$

We define a set of margins associated with a training example as

$$\rho_{i,r} = H_{i:}\boldsymbol{w}_{y_i} - H_{i:}\boldsymbol{w}_r,\ r = 1, \ldots, k. \tag{11}$$

Clearly only when $\rho_{i,r} \geq 0$, will the training example $\boldsymbol{x}_i$ be correctly classified. We consider the logarithmic version of the original cost function, which does not change the problem because $\log(\cdot)$ is strictly monotonically increasing. So we write (10) into

$$\begin{aligned} &\min_{W,\boldsymbol{\rho}} \log\Big(\sum_{i=1}^{m}\sum_{r=1}^{k} \exp\left[-\rho_{i,r}\right]\Big) + \nu \left\|W\right\|_1 \\ &\text{s.t. } \rho_{i,r} = H_{i:}\boldsymbol{w}_{y_i} - H_{i:}\boldsymbol{w}_r, \forall i = 1 \cdots m, r = 1 \cdots k, \\ &\qquad W \geq 0. \end{aligned} \tag{12}$$

The dual problem can be easily derived:

$$\begin{aligned} &\min_{U} \sum_{r=1}^{k}\sum_{i=1}^{m} U_{ir} \log U_{ir} \\ &\text{s.t. } \sum_{i} \left[\delta_{r,y_i}\left(\textstyle\sum_{l=1}^{k} U_{il}\right) - U_{ir}\right] H_{i:} \leq \nu \mathbf{1}^\top, \forall r, \\ &\qquad \sum_{i,r} U_{ir} = 1, U \geq 0. \end{aligned} \tag{13}$$

We can see that the dual problem is a Shanon entropy maximization problem. The objective function of the dual encourages the weights U to be uniform. The KKT condition gives the relationship between the optimal primal and dual variables:

$$U^\star_{ir} = \frac{\exp(-\rho^\star_{i,r})}{\sum_{i,r}\exp(-\rho^\star_{i,r})},\ \forall i, r. \tag{14}$$

Different from the case of the hinge loss, here U is normalized as an entire matrix. Also we can solve the primal problem using simple (Quasi-)Newton, which is much faster than to solve the dual problem using convex optimization solvers. Note that the scale of the primal problem is usually smaller than the dual problem. After obtaining the primal variable, we can use the KKT condition to get the dual variable. The subproblem that we need to solve for generating weak classifiers also slightly differs from (8):

$$h^*(\cdot) = \operatorname*{argmax}_{h(\cdot)} \sum_{i=1}^{m} \Big(\delta_{r,y_i}\left(\textstyle\sum_{l=1}^{k} U_{il}\right) - U_{ir}\Big) h(\boldsymbol{x}_i). \tag{15}$$

2.3. General convex loss

We generalize the presented idea to *any smooth convex* loss functions in this section. Suppose $\lambda(\cdot)$ is a smooth convex function defined in $\mathbb{R}$. For classification problems, $\lambda(\cdot)$ is usually a convex surrogate of the non-convex zero-one loss. As in the exponential loss case, we introduce a set auxiliary variables that define the margin as the pairwise difference of prediction scores. This auxiliary variable is the key to lead to the important Lagrange dual, on which the totally corrective boosting algorithms rely.

The optimization problem can be formulated as

$$\min_{W,\boldsymbol{\rho}} \sum_{i=1}^{m}\sum_{r=1}^{k} \lambda(-\rho_{i,r}) + \nu \left\|W\right\|_1 \text{ s.t. (11), and } W \geq 0. \tag{16}$$

The Lagrangian is

$$\begin{aligned} L = &\sum_{i,r} \lambda(-\rho_{i,r}) - \mathbf{Tr}(V^\top W) + \sum_{i,r} U_{ir}(H_{i:}\boldsymbol{w}_{y_i} - H_{i:}\boldsymbol{w}_r) \\ &- \sum_{i,r} U_{ir}\rho_{i,r} + \nu \sum_{j,r} W_{jr}. \end{aligned}$$

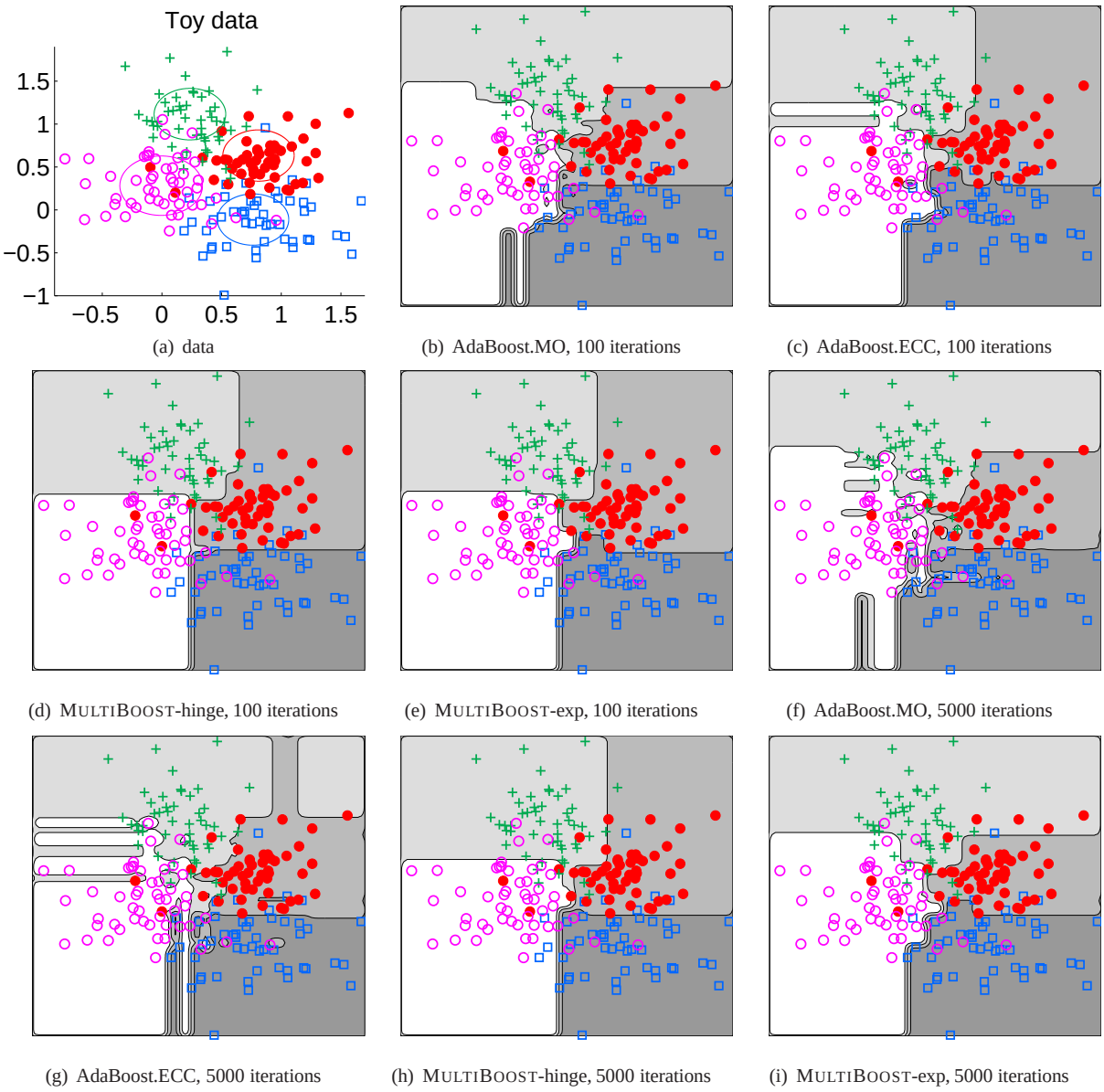
We can again write its Lagrange dual as

$$\begin{aligned} &\min_{U} \sum_{r=1}^{k}\sum_{i=1}^{m} \lambda^*(-U_{ir}) \\ &\text{s.t. } \sum_{i} \left[\delta_{r,y_i}\left(\textstyle\sum_{l=1}^{k} U_{il}\right) - U_{ir}\right] H_{i:} \leq \nu \mathbf{1}^\top, \forall r, \end{aligned} \tag{17}$$

where $\lambda^*(\cdot)$ is the Fenchel dual function of $\lambda(\cdot)$ [2]. Note that $\lambda^*(\cdot)$ is always convex even if the original loss function $\lambda(\cdot)$ is non-convex. The difference is that the duality gap is not zero when $\lambda(\cdot)$ is non-convex. The KKT condition establishes the connection between the dual variable U and the primal variable at optimality:

$$U_{ir} = -\lambda'(\rho_{i,r}). \tag{18}$$

So we can actually solve the primal problem and then recover the dual solution from the primal. From (18), we



(a) data (b) AdaBoost.MO, 100 iterations (c) AdaBoost.ECC, 100 iterations

(d) MULTIBOOST-hinge, 100 iterations (e) MULTIBOOST-exp, 100 iterations (f) AdaBoost.MO, 5000 iterations

(g) AdaBoost.ECC, 5000 iterations (h) MULTIBOOST-hinge, 5000 iterations (i) MULTIBOOST-exp, 5000 iterations

Figure 1: Figure (a) shows a toy data set, which contains 4 classes and a total of 200 sample points. Boosting algorithms are trained on this set using decision stumps. Plots (b)-(e) illustrate the decision boundaries made by (b) AdaBoost.MO, (c) AdaBoost.ECC, (d) MULTIBOOST-hinge and (e) MULTIBOOST-exp with the number of training iterations being 100. For comparison, plots (f)-(i) illustrate the decision boundaries of these algorithms, respectively, when the number of iterations is 5000. (f) and (g) apparently suffer from over-fitting.

know that the weight U is typically non-negative for classification problems because the classification loss function $\lambda(\cdot)$ is monotonically decreasing and its gradient is non-positive.

3. Experiments

We have performed a few sets of experiments to compare the boosting algorithms that we proposed with previous multi-class boosting algorithms. For fair comparison, we focus on the multi-class algorithms using binary weak learners, including AdaBoost.MO and AdaBoost.ECC, which are still considered as the state-of-the-

dataset	AdaBoost.MO	AdaBoost.ECC	MULTIBOOST-hinge	MULTIBOOST-exp
thyroid	0.005±0.001	0.005±0.001	0.005±0.001	**0.004±0.001**
dna	0.059±0.005	0.064±0.005	**0.057±0.007**	0.061±0.004
wine	0.036±0.025	0.034±0.029	0.032±0.018	**0.030±0.029**
iris	0.062±0.017	0.073±0.021	0.068±0.022	**0.057±0.022**
glass	**0.232±0.047**	0.242±0.053	0.234±0.046	0.315±0.086
svmguide2	0.213±0.039	0.214±0.030	0.222±0.052	**0.206±0.040**
svmguide4	0.192±0.018	**0.191±0.018**	0.207±0.018	0.214±0.027

Table 1: Test errors of four boosting algorithms on UCI data sets. The average results of 10 repeated tests are reported. Weak classifiers are decision stumps. MULTIBOOST-exp is the best on 4 out of 7 data sets.

art.

For AdaBoost.MO, the error-correcting output codes are introduced to reduce the primal problem into multiple binary ones; for AdaBoost.ECC, the binary partitioning is made at each iteration by using the "random-half" method, which has been experimentally proven better than the optimal "max-cut" solution [10]. Decision stumps are chosen as the weak classifiers for all boosting algorithms, due to its simplicity and the controlled complexity of the weak learner.

Convex optimization problems are involved in MULTIBOOST-hinge and MULTIBOOST-exp. To solve them, we use the off-the-shelf Mosek convex optimization package [1], which provides solutions for both primal and dual problems simultaneously with its interior-point Newton method.

We also need to set the regularization parameter ν for these two algorithms using cross validation. For each run, a five-fold cross validation is carried out first to determine the best ν. Notice that the loss functions in MULTIBOOST-hinge and MULTIBOOST-exp may have different scales, we choose the parameter from $\{10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, 0.01, 0.02, 0.04, 0.05\}$ for the former, and the candidate pool $\{10^{-8}, 10^{-7}, 5 \cdot 10^{-7}, 8 \cdot 10^{-7}, 10^{-6}, 2 \cdot 10^{-6}, 4 \cdot 10^{-6}, 8 \cdot 10^{-6}, 10^{-5}\}$ for the latter.

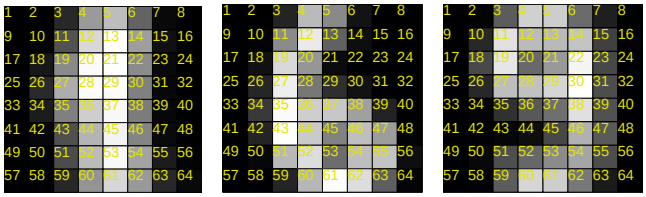
Toy data In the first experiment, we make the comparison on a toy data set, which consists of 4 clusters of planar points. Each cluster has 50 samples, which are drawn from their respective normal distribution. As shown in Figure 1(a), the centers of the circles indicate where their means are, and the radii depict the different deviations. We run the boosting algorithms on this toy data set and plot the decision boundaries on the x-y plane. Figures 1(b)-(e) illustrate the results when the number of training iterations is set to be 100. In this case, it is hard to state which model is better. However, if we increase the iteration to 5000 times, the planes in (f) and (g) are apparently over segmented by AdaBoost.MO and AdaBoost.ECC. On the contrary, the decision boundaries of (h) MULTIBOOST-hinge and (i) MULTIBOOST-exp seem closer to the true decision boundary. Unlike the others, models trained by AdaBoost.MO are more complex, since this learning method assembles ℓ weak classifiers rather than one at each iteration if ℓ-length codewords are used. Empirically we see that AdaBoost.ECC also seems susceptible to over-fitting.

UCI data sets Next we test our algorithms on 7 data sets collected from UCI repository. Samples are randomly divided into 75% for training and 25% for test, no matter whether there is a pre-specified split or not. Each data set is run 10 times and the average results of test error are reported in Table 1. The maximum number of iterations is set to 500. Almost all the algorithms converge before the maximum iteration. Again the regularization parameter is determined by 5-fold cross validation.

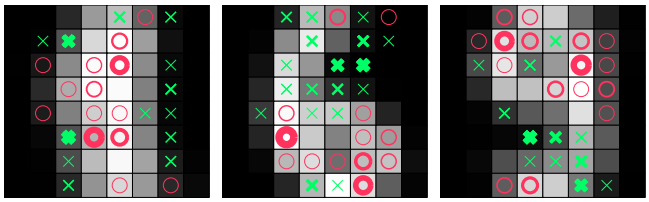
Table 1 reports the results. The conclusion that we can draw on this experiment is: 1) Overall, all the algorithms achieve comparable accuracy. 2) our algorithms are slightly better in terms of generalization ability than the other two on 5 out of 7 data sets. MULTIBOOST-exp outperforms others in 4 data sets. 3) Also note that the performance MULTIBOOST-hinge is more stable than MULTIBOOST-exp, which may be due to the fact that the hinge loss is less sensitive to noise than the exponential loss.

Handwritten digits recognition To further examine the effectiveness of our algorithms, We have conducted another experiment on a handwritten digits data set, which is also from UCI repository. The original data set contains 5620 digits written by a total of 43 people on 32×32 bitmaps. Then the bitmaps are divided into 4×4 non-overlapping blocks, and an 8×8 descriptor is generated by calculating the sum of 0-1 pixels in each block. For ease of exposition, only 3 distinct digits of "1", "6" and "9" are chosen for classification. Figure 2(a) illustrates the mean images of their training data examples of the three digits. The index of each block (feature) is also printed on Figure 2(a) for the convenience of exposition.

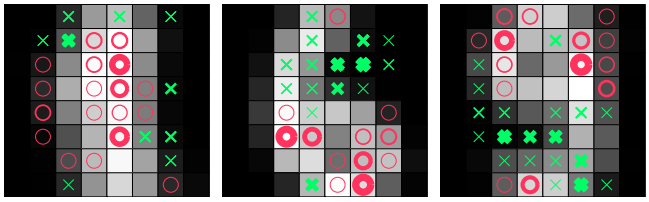
We train multi-class boosting on this data set. The number of maximum training iterations is set to 500. 75% data are used for training, and the rest for test. Again 5-fold cross validation is used. We still use decision stumps as the weak classifiers. Boosting learning with decision stumps implies



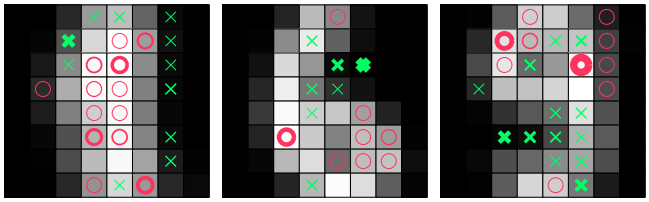
(a) mean images of "1", "6" and "9"



(b) AdaBoost.ECC



(c) MULTIBOOST-hinge



(d) MULTIBOOST-exp

Figure 2: Plot (a) shows the mean images of the samples belonging to digits "1", "6" and "9". Each block is a feature and is numerically indexed. The remaining plots illustrate the classification models trained on this data set by (b) AdaBoost.ECC, (c) MULTIBOOST-hinge and (d) MULTIBOOST-exp. Red circles indicate that weak classifiers on these features should take large values; Green crosses indicate small values should be taken on these features in order to make correct classification. The width of a mark is proportional to the weight of the stump. We can see that MULTIBOOST-hinge is slightly better than AdaBoost.ECC, e.g., on the 43-th and 21-th features.

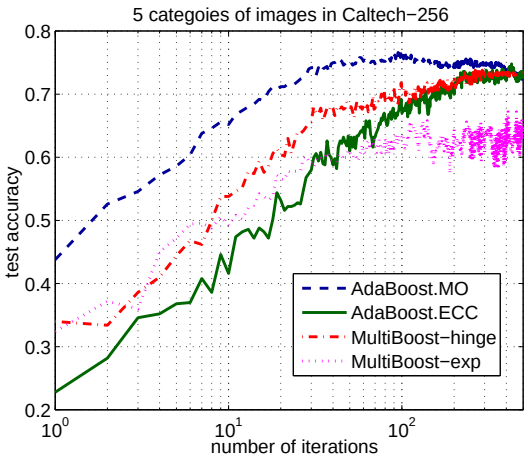


Figure 3: Test accuracy curves of four boosting algorithms on 5 categories of Caltech-256 images. The weak classifiers are decision stumps and the number of training iterations is 500. The average results of 10 runs are reported. Each run randomly selects 75% data for training and the other 25% for test.

that we select features at the same time. In other words, decision stumps select most discriminative blocks for classifying these digits. The four compared algorithms have similar performances on this test with nearly 98% test accuracy. We plot the models of AdaBoost.ECC, MULTIBOOST-hinge and MULTIBOOST-exp in Figures 2(b)-(d). AdaBoost.MO can be hardly illustrated as it involves a multi-dimensional coding scheme. Notice that a decision stump divides the value range of the feature into two parts, on which there are necessarily two different attributions, we use red circles and green crosses to represent the positive and negative parts. For example, if a decision stump on the 10-th feature is $x_{10} > \tau$ and assigns a set of weights $\{0.5, 0.2, 0.8\}$ to three labels, we mark 10-th block in the third digit image with a red circle, and 10-th block in the second digit with a green cross; if the stump is $x_{10} < \tau$ with the same weights, we do the opposite marks. In other words, red circles indicate the decision stumps should take bigger values on these blocks, while green crosses indicate these classifiers should take some values as small as possible. The width of a mark stands for the minimal margin defined in Equation (11), that is, in the i-th digit, the width is proportional to $h(\boldsymbol{x})\boldsymbol{w}_{y_i} - \max\{h(\boldsymbol{x})\boldsymbol{w}_r\}, \forall r \neq y_i$. Some features may be selected multiple times, which divide the value range into several segments. In this case, we neglect all the middle parts.

Clearly, all the results of three algorithms on feature selection make sense. Most discriminative features are tagged with circles or crosses. Some blocks that contain significant information on luminance are tagged with thick marks, such as the 22-th and 43-th features in digit "6", and the 22-th and 11-th in "9". If taking a close look at the figure, we can find MULTIBOOST-hinge is slightly better than AdaBoost.ECC. For example, on the 43-th feature the green cross should be

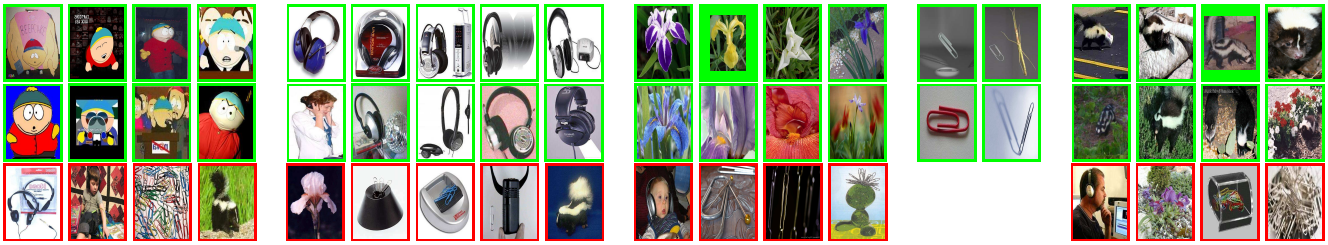


Figure 4: Some examples of correctly classified (top two rows) and misclassified (bottom row) images by MULTIBOOST--hinge. The categories are "cartman", "headphones", "iris", "paperclip" and "skunk". The accuracy of this test is 71.2%. No image is falsely classified into the category of "paperclip".

marked on digit "9" instead of "1". Also in "1", the 21-th feature should be tagged with a relatively thicker circle. However, MULTIBOOST-exp's results are not as meaningful as MULTIBOOST-hinge.

Object recognition on a subset of Caltech-256 Finally, we test our algorithms on the data set of Caltech-256, which is one of the most popular multi-class benchmarks. We randomly select 5 categories of images. 75% of them are randomly selected for training and the other 25% for test. A descriptor of 1000 dimensions is used, which combines quantized color and texture local invariant features (also called *visterms* [12]). The maximum number of iterations is still set to 500. The averaged test accuracies of 10 runs are reported in Figure 3. Again, we use the simplest decision stumps as weak classifiers. We can see that all the four boosting algorithms perform similarly, except that MULTIBOOST-exp performs worse than the other three. It may be due to the fact that we have not fine tuned the cross validation parameter. We show some images correctly classified and falsely classified by MULTIBOOST-hinge in Figure 4.

4. Conclusion

In this work, we have presented a direct formulation for multi-class boosting. We derive the Lagrange dual of the formulated primal optimization problem. Based on the dual problem, we are able to design totally-corrective boosting using the column generation technique. At each iteration, all weak classifiers' weights are updated.

Various experiments on a few different data sets demonstrate that our direct multi-class boosting achieves competitive test accuracy compared with other existing multi-class boosting.

Future research topics include how to efficiently solve the convex optimization problems of the proposed multi-class boosting. Conventional multi-class boosting do not need to solve convex optimization at each step and thus much faster. We also want to explore the possibility of structural learning with boosting by extending the proposed multi-class boosting framework.

References

[1] The MOSEK optimization tools manual (version 6.0).

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] K. Crammer and Y. Singer. On the algorithmic implemtation of multiclass kernel-based vector machines. *J. Mach. Learn. Research*, 2:265–292, 2001.

[4] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Mach. Learn.*, 47(2):201–233, 2002.

[5] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Mach. Learn.*, 46(1–3):225–254, 2002.

[6] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artificial Intelligence Research*, 2:263–286, 1995.

[7] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *Proc. Adv. Neural Info. Process. Syst.*, pages 681–687. MIT Press, 2001.

[8] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Computer & System Sciences*, 55(1):119–139, 1997.

[9] V. Guruswami and A. Sahai. Multiclass learning, boosting, and errorcorrecting codes. In *Proc. Annual Conf. Learn. Theory*, pages 145–155, 1999.

[10] L. Li. Multiclass boosting with repartitioning. In *Proc. Int. Conf. Mach. Learn.*, pages 569–576, 2006.

[11] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Proc. Adv. Neural Info. Process. Syst.*, pages 512–518. MIT Press, 2000.

[12] P. Quelhas and J. M. Odobez. Natural scene image modeling using color and texture visterms. *Image & Video Retrieval*, pages 411–421, 2006.

[13] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Ann. Statist.*, 26(5):1651–1686, 1998.

[14] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Mach.*

Learn., 37(3):297–336, 1999.

[15] R. E. Schapire. Using output codes to boost multiclass learning problems. In *Proc. Int. Conf. Mach. Learn.*, pages 313–321, 1997.

[16] C. Shen and H. Li. On the dual formulation of boosting algorithms. *IEEE Trans. Pattern Anal. & Mach. Intelligence*, 32(12):2216–2231, 2010.

[17] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Research*, 6:1453–1484, 2005.

[18] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comp. Vis.*, 57(2):137–154, 2004.

[19] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proc. Euro. Symp. Artificial Neural Networks*, volume 4, pages 219–224, 1999.

[20] J. Zhu, S. Rosset, H. Zou, and T. Hastie. Multi-class AdaBoost. *Stat. & its interface*, 2:349–360, 2009.