Sharing Features in Multi-class Boosting via Group Sparsity

Sakrapee Paisitkriangkrai, Chunhua Shen, Anton van den Hengel
The Australian Centre for Visual Technologies and School of Computer Science
The University of Adelaide, Australia

Abstract

We present a novel formulation of fully corrective boosting for multi-class classification problems with the awareness of sharing features. Our multi-class boosting is solved in a single optimization problem. In order to share features across different classes, we introduce the mixed-norm regularization, which promotes group sparsity, into boosting. We then derive the Lagrange dual problems which enable us to design fully corrective multi-class algorithms using the primal-dual optimization technique. We show that sharing features across classes can improve classification performance and efficiency. We empirically show that in many cases, the proposed multi-class boosting generalizes better than a range of competing multi-class boosting algorithms due to the capability of feature sharing. Experimental results on machine learning data, visual scene and object recognition demonstrate the efficiency and effectiveness of proposed algorithms and validate our theoretical findings.

1. Introduction

A significant proportion of the most important classification problems inherently exhibit a large number of classes. These problems demand effective and efficient multi-class classification techniques. Unlike binary classification, which has been well researched, multi-class classification has received relatively little attention due to the inherent complexity of the problem. Some important steps have been taken towards addressing the problem (see [8,16,28] for instance), but the primary approach taken thus far has exploited large numbers of independent binary classifiers. An example of this approach is the extension of a binary classification algorithm to the multi-class case by considering the problem as a bunch of one-vs-all binary classification problem. The disadvantage of this approach is that the final ensemble classifier is often made up of a large number of weak classifiers which inevitably leads to long evaluation times.

We propose here a novel formulation of the fully corrective boosting for multi-class classification problems with the awareness of feature sharing[1]. Boosting is an ensemble classifier learning technique. It combines a set of weak classifiers, which are generated by a base learning oracle, in order to form a strong classifier. Recently, boosting has attracted much research interest in the machine learning and pattern recognition community due to its robustness and efficiency [12]. The key intuition behind our work is that informative features are commonly shared between various classes. For example, traffic warning signs have a common triangular shape with various symbols inside. These basic shared features can be used to help differentiate warning signs from other objects while the symbols inside can be used to differentiate different warning signs. In this work, we aim to select a common subset of features which are informative in identifying a wide range of classes for a multi-class problem.

Main contributions 1) We propose a new formulation for multi-class boosting that promotes feature sharing across classes by enforcing group sparsity regularization (referred to as MultiBoost$^{\text{group}}$). Group sparsity regularization can be extremely useful in problems where there exist a structure over sample features, *e.g.*, feature sharing, and when features are expensive to compute. We empirically show that by enforcing group sparsity, the proposed multi-class boosting converges faster while achieving better or comparable generalization performance. The fact that the algorithm converges fast means that fewer features are required for a given classification accuracy and there is a significant improvement in run-time performance. Our derivation for designing multi-class boosting methods is applicable to general $\ell_{p,q}$ $(p, q \geq 1)$ mixed-norms. We also propose the use of the alternating direction method of multipliers (ADMM) [4] to efficiently solve the involved optimization problems, which is much faster than using standard interior-point solvers such as MOSEK [1]. *To our knowledge, this is the first fully-corrective multi-class boosting approach that promotes feature sharing using group sparsity regularization.* 2) We propose a new family of multi-

[1]We use "features" and "weak classifiers" interchangeably when the weak classifier is a decision stump because a decision stump is trained on a single feature of the input data.

978-1-4673-1228-8/12/$31.00 ©2012 IEEE

class boosting algorithms based on a simplified formulation. This formulation not only enables us to share features and encourages structural sparsity in the learning procedure of multi-class boosting, but also allows us to take advantage of parallelism in ADMM to speed up the training time by a factor proportional to the number of classes k. The training time required is thus similar to that required to train multiple independent binary classifiers in parallel. The proposed formulation converges significantly faster, however, by virtue of the fact that features may be shared between classes, thus exploiting group sparsity.

Related work Boosting is a well-known technique commonly applied to improve the accuracy of a learning procedure. The algorithm forms an ensemble classifier from a weighted combination of weak, or base, learners. The final boosted strong classifier is capable of achieving high classification accuracy. Boosting was originally proposed for binary classification [13]. It has then been extended to multi-class problems [2,14,22]. Multi-class boosting can be achieved through an ensemble of 1) multi-class weak learners [14,30] or 2) binary weak learners [2,22]. In the former, the weak learner is required to produce a multi-class output, while in the latter, the weak learner is only required to product a binary output. Here we focus on the latter. The advantage of forming an ensemble using binary weak classifiers is that the binary classification problem has been well studied and many effective algorithms have been designed for binary problems. Moreover, binary weak learners are often much simpler and more efficient than multi-class weak learners. As a result, they are often faster to train and have a better generalization ability (less likely to over-fit the training data).

One well-known approach to build a multi-class classifier is to use a coding matrix to reduce the output space of a multi-class problem into that of several binary problems. Some examples of such strategies include one-vs-all [21], one-vs-one (round robin classification) [15] and error-correcting output coding [8, 10]. Multi-class boosting algorithms exploiting these strategies include AdaBoost.MH, AdaBoost.MO, AdaBoost.OC, AdaBoost.ECC, among numerous others. Unfortunately, all of these algorithms fail to consider similarity between classes, especially the feature sharing property. Since binary classifiers are trained independently, the resulting strong classifier can be highly unbalanced and often dependent on an excessive number of features/weak classifiers.

Several work has been introduced to address the feature sharing problem in multi-class boosting learning. JointBoost, proposed by Torralba *et al*. [26], finds common features that can be shared across classes using heuristics. Weak learners are then trained jointly using standard boosting. Zhang *et al*. proposed to train multi-class boosting with sharable information patterns [29]. As a pre-processing step, they generate sharable patterns using data mining techniques and then learn a multi-class boosting on these patterns. So the finding of sharable features and multi-boost training are de-coupled.

In comparison to JointBoost and Zhang *et al*.'s work, we select weak learners systematically on the basis of structural sparsity. A related approach is a multi-class boosting of Duchi and Singer [11] known as GradBoost, which has also used mixed-norm for group sparsity. The main difference is that GradBoost of [11] does not directly optimize the boosting objective function. Instead, the algorithm updates a block of variables for optimizing a quadratic surrogate function, in a fashion similar to gradient-based coordinate descent. It is not clear how well the surrogate approximates the original objective function. Since the mixed-norm regularization is not directly optimized either, *there the group sparsity is achieved heuristically by a combination of forward selection and backward elimination*. Our work fundamentally differs [11] in that we directly optimize the group sparsity regularized objective by following the column generation based boosting [23] and no heuristics is involved in the optimization. Shen and Hao [23] introduced a direct formulation for multi-class boosting. But feature sharing is not considered in their work. The work of [23] can be seen as an extension of the column generation boosting framework of [24] to multi-class. Here we design our feature-sharing multi-class boosting in a direct formulation as well, but with more sophisticated group sparsity regularization. Note that the general boosting framework of [23, 24] is not directly applicable in our problem setting.

Notation Let $(\boldsymbol{x}_i, y_i)_{i=1}^m$ be the set of training data, where $\boldsymbol{x}_i \in \mathbb{R}^D$ represents an training example, and $y_i \in \{1, 2, \cdots, k\}$ the corresponding class label. We have m training samples and k classes. Let $h(\cdot)$ be a weak classifier which projects an input vector $\boldsymbol{x}$ into $\{-1, +1\}$ (here we consider only binary classifiers although the proposed approach can be applied to any real-valued weak classifiers). We define the matrix $H \in \mathbb{Z}^{m\times n}$, which is made up of the binary outputs of the weak classifiers when applied to the training samples. So the (i, j) entry of H, $H_{ij} = h_j(\boldsymbol{x}_i)$, is the label predicted by weak classifier $h_j(\cdot)$ for the datum $\boldsymbol{x}_i$. Each column $H_{:j}$ of the matrix H thus represents the output of a single weak classifier $h_j(\cdot)$, and each row $H_{i:}$ the output of all weak classifiers when applied to a single training datum $\boldsymbol{x}_i$. In this work, we aim to learn a linear ensemble classifier $\sum_{j=1}^n w_j h_j(\cdot)$ for each class. Here $w_1, w_2, \cdots, w_n$ are the coefficients of the linear classifier. Since we have k classes, we define the matrix $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_k] \in \mathbb{R}^{n\times k}$ such that each column of W, $\boldsymbol{w}_r$, contains coefficients of the linear classifier for class r and each row of W, $W_{j:}$, consists of the coefficients for the weak classifier $h_j(\cdot)$ for all class labels. The $\ell_{1,2}$ norm of a matrix is defined as $\|W\|_{1,2} = \sum_j \|W_{j:}\|_2$ with $\|\cdot\|_2$ being the ℓ_2 norm. The fi-

nal strong classifier is a weighted average of multiple weak classifiers, and the estimated classification for a test datum $\boldsymbol{x}$ is $F(\boldsymbol{x}) = \underset{\ell}{\operatorname{argmax}} \sum_{j=1}^{n} W_{j\ell} h_j(\boldsymbol{x})$.

2. Multi-class boosting with group sparsity

In this section we formulate the multi-class boosting algorithm using mixed norm regularization, and a variety of loss criteria. The fact that the classifier is based on pairwise comparisons between classes means that the response of the linear classifiers corresponding to the correct label must be larger than that of the linear classifiers representing other labels. Let $F_{y_i}(\cdot)$ be the response of the linear classifier corresponding to y_i (the true class label) when applied to training instance $\boldsymbol{x}_i$. The multi-class margins for the instance $\boldsymbol{x}_i$ thus can be defined as $F_{y_i}(\boldsymbol{x}_i) - F_r(\boldsymbol{x}_i), \forall r \neq y_i$.

Multi-class hinge loss In order to maximize the margin using the hinge loss, the following conditions are encouraged to be satisfied, $F_{y_i}(\boldsymbol{x}_i) \geq 1 + F_r(\boldsymbol{x}_i), \forall r \neq y_i$. The condition states that the confidence of the correct label should be larger than the confidence of other labels by at least one unit. 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

$$\delta_{r,y_i} + F_{y_i}(\boldsymbol{x}_i) \geq 1 + F_r(\boldsymbol{x}_i), r = 1, 2, \cdots, k.$$

Given training samples, our goal is to minimize the multi-class hinge loss with $\ell_{1,2}$ mixed-norm regularization. The primal problem can be written as

$$\begin{aligned} \min_{W,\boldsymbol{\xi}} \; & \sum_{i=1}^{m} \xi_i + \nu \|W\|_{1,2} \\ \text{s.t. } & \delta_{r,y_i} + H_{i:}\boldsymbol{w}_{y_i} \geq 1 + H_{i:}\boldsymbol{w}_r - \xi_i, \forall i, r; \\ & W \geq 0; \boldsymbol{\xi} \geq 0. \end{aligned} \tag{1}$$

Here $\nu > 0$ is the regularization parameter. We rewrite (1) by introducing an auxiliary variable V:

$$\begin{aligned} \min_{W,V,\boldsymbol{\xi}} \; & \sum_{i=1}^{m} \xi_i + \nu \|V\|_{1,2} \\ \text{s.t. } & \delta_{r,y_i} + H_{i:}\boldsymbol{w}_{y_i} \geq 1 + H_{i:}\boldsymbol{w}_r - \xi_i, \forall i, r \\ & V = W; W \geq 0; \boldsymbol{\xi} \geq 0. \end{aligned} \tag{2}$$

This auxiliary variable V splits the regularization term from the classification loss, and plays a critical role in deriving the meaningful dual problem. Actually $\boldsymbol{\xi} \geq 0$ is automatically satisfied since the constraint, corresponding to the case $r = y_i$, ensures the non-negativeness of $\boldsymbol{\xi}$. We derive its Lagrange dual, as in LPBoost [9]. The Lagrangian can then be written as

$$\begin{aligned} L = & \sum_{i=1}^{m} \xi_i + \nu \|V\|_{1,2} - \sum_{i,r} U_{ir}(\delta_{r,y_i} + H_{i:}\boldsymbol{w}_{y_i} - 1 \\ & - H_{i:}\boldsymbol{w}_r + \xi_i) - \langle Q, \nu W - \nu V \rangle - \langle P, W \rangle, \end{aligned}$$

where W, V and $\boldsymbol{\xi}$ are primal variables and U, P and Q are dual variables (with $U \geq 0$ and $P \geq 0$). At optimum, the first derivative of the Lagrangian w.r.t. the primal variables, $\boldsymbol{\xi}$, must vanish, $\partial L / \partial \xi_i = 0 \rightarrow \sum_r U_{ir} = 1, \forall i$. The first derivative w.r.t. each column of W must also be zeros:

$$\begin{aligned} & \frac{\partial L}{\partial \boldsymbol{w}_r} = \mathbf{0} \\ & \rightarrow \sum_i U_{ir} H_{i:} - \sum_{i, r=y_i} \underbrace{\left[\sum_l U_{il}\right]}_{=1} H_{i:} = P_{:r} - \nu Q_{:r} \\ & \rightarrow \textstyle\sum_i U_{ir} H_{i:} - \sum_i \delta_{r,y_i} H_{i:} \geq -\nu Q_{:r}. \end{aligned} \tag{3}$$

The infimum over the primal variables V can be expressed as

$$\begin{aligned} \inf_V L &= \inf_V -\nu \langle Q, V \rangle + \nu \|V\|_{1,2} \\ &= -\nu \textstyle\sum_j \left[\sup_{V_{j:}} Q_{j:}^\top V_{j:} - \|V_{j:}\|_2\right] \\ &= -\nu \textstyle\sum_j \begin{cases} 0 & \text{if } \|Q_{j:}\|_2 \leq 1, \forall j, \\ \infty & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

Note that we use the fact that the convex conjugate of $\|V_{j:}\|_2$ is the indicator function of the dual norm unit ball [5]. Hence the Lagrange dual can be written as

$$\begin{aligned} \min_{U,Q} \; & \sum_{i,r} U_{ir} \delta_{r,y_i} \\ \text{s.t. } & \textstyle\sum_i (\delta_{r,y_i} - U_{ir}) H_{i:} \leq \nu Q_{:r}, \forall r; \\ & \textstyle\sum_r U_{ir} = 1, \forall i; \; U \geq 0; \quad \|Q_{j:}\|_2 \leq 1, \forall j. \end{aligned} \tag{5}$$

Since there can be infinitely many constraints, we need to use column generation to solve (5) [9]. The subproblem for generating weak classifiers is

$$h^*(\cdot) = \underset{h(\cdot) \in \mathcal{H}, r}{\operatorname{argmax}} \sum_{i=1}^{m} (\delta_{r,y_i} - U_{ir}) h(\boldsymbol{x}_i). \tag{6}$$

$h^*(\cdot)$ is the one that most violates the first constraint in the dual (5). The idea of column generation is that instead of solving the original problem with prohibitively large number of constraints, we consider instead a small subset of entire variable sets. The algorithm begins by finding a variable that most violates the dual constraints, *i.e.*, the solution to (6), which corresponds inserting a primal variable into (1) or (2). The process continues as long as there exists at least one constraint that is violated for (5). The algorithm terminates when we cannot find such a violated constraint. As in AdaBoost, the matrix $U \in \mathbb{R}^{m \times k}$ plays the role of measuring the importance of the training samples. The weak classifier which maximizes (6) is selected in each iteration. Similarly, the $\ell_{1,\infty}$-norm regularized primal can be written as

$$\begin{aligned} \min_{W,V,\boldsymbol{\xi}} \; & \sum_{i=1}^{m} \xi_i + \nu \|V\|_{1,\infty} \\ \text{s.t. } & \delta_{r,y_i} + H_{i:}\boldsymbol{w}_{y_i} \geq 1 + H_{i:}\boldsymbol{w}_r - \xi_i, \forall i, r \\ & V = W; W \geq 0; \quad \boldsymbol{\xi} \geq 0. \end{aligned} \tag{7}$$

Its corresponding dual is

$$\min_{U,Q} \quad \sum_{i,r} U_{ir}\delta_{r,y_i} \tag{8}$$
$$\text{s.t.} \quad \textstyle\sum_i(\delta_{r,y_i} - U_{ir})H_{ij} \le \nu Q_{:r}, \forall r;$$
$$\textstyle\sum_r U_{ir} = 1, \forall i; U \ge 0; \quad \|Q_{j:}\|_1 \le 1, \forall j.$$

From the dual problem we see that the only difference between $\ell_{1,2}$-norms and $\ell_{1,\infty}$-norms is in the norm of the last constraint. This is not surprising since ℓ_p norm in primal corresponds to ℓ_q norm in dual with $1/p + 1/q = 1$.

Multi-class logistic loss We can also design a boosting algorithm for optimizing the logistic loss function:

$$\frac{1}{mk}\sum_{i=1}^{m}\sum_{r=1}^{k}\log\big(1+\exp\left(H_{i:}\boldsymbol{w}_r - H_{i:}\boldsymbol{w}_{y_i}\right)\big). \tag{9}$$

As in the previous derivation, we put the above logistic loss in an $\ell_{1,2}$ regularization framework. The learning problem can then be expressed as,

$$\min_{W,V,\boldsymbol{\rho}} \frac{1}{mk}\sum_{i=1}^{m}\sum_{r=1}^{k}\log\big(1+\exp\left(-\rho_{ir}\right)\big) + \nu\|V\|_{1,2} \tag{10}$$
$$\text{s.t. } \rho_{ir} = H_{i:}\boldsymbol{w}_{y_i} - H_{i:}\boldsymbol{w}_r, \forall i, \forall r,$$
$$V = W; W \ge 0.$$

Here we introduce the auxiliary variables, $\boldsymbol{\rho}$, and additional constraints, $V = W$, to obtain the meaningful dual formulation. The Lagrange dual can be written as (see the supplementary for details):

$$\max_{U,Q} \quad -\frac{1}{mk}\sum_{i=1}^{m}\sum_{r=1}^{k}\Big[mkU_{ir}\log\left(mkU_{ir}\right) + \tag{11}$$
$$\left(1-mkU_{ir}\right)\log\left(1-mkU_{ir}\right)\Big]$$
$$\text{s.t.} \quad \textstyle\sum_i\big[\delta_{r,y_i}\left(\sum_l U_{il}\right) - U_{ir}\big]H_{i:} \le \nu Q_{:r}, \forall r;$$
$$\|Q_{j:}\|_2 \le 1, \forall j.$$

Through the Karush-Kunh-Tucker (KKT) optimality condition, the gradient of Lagrangian over primal variables $\boldsymbol{\rho}$ and dual variables U must vanish at the optimum. The solutions of (10) and (11) coincide since both problems are feasible and satisfy Slater's condition. One can find the solution by solving either problem. The relationship between the optimal values of $\boldsymbol{\rho}$ and U can be expressed as

$$U_{ir} = \frac{\exp(-\rho_{ir})}{mk\big(1+\exp(-\rho_{ir})\big)}. \tag{12}$$

As was the case for the hinge loss, the dual of the $\ell_{1,\infty}$-norm regularized logistic loss can be written as

$$\max_{U,Q} \quad -\frac{1}{mk}\sum_{i=1}^{m}\sum_{r=1}^{k}\Big[mkU_{ir}\log\left(mkU_{ir}\right) + \tag{13}$$
$$\left(1-mkU_{ir}\right)\log\left(1-mkU_{ir}\right)\Big]$$
$$\text{s.t.} \quad \textstyle\sum_i\big[\delta_{r,y_i}\left(\sum_l U_{il}\right) - U_{ir}\big]H_{i:} \le \nu Q_{:r}, \forall r;$$
$$\|Q_{j:}\|_1 \le 1, \forall j.$$

Algorithm 1 MultiBoost with shared weak classifiers via group sparsity.

Input:
1) A set of examples $\{\boldsymbol{x}_i, y_i\}, i = 1\cdots m$; 2) The maximum number of weak classifiers, T;
Output: A multi-class classifier $F(\boldsymbol{x}) = \operatorname{argmax}_r \sum_{j=1}^{T} W_{jr}h_j(\boldsymbol{x})$;
Initilaize:
1) $t \leftarrow 0$; 2) Initialize sample weights, $U_{ir} = 1/(mk)$;
while $t < T$ **do**
  1) Train a weak learner, $h_t(\cdot) =$

$$\begin{cases} \operatorname{argmax}_{h(\cdot),r} \sum_{i=1}^{m}\left[\delta_{r,y_i} - U_{ir}\right]h(\boldsymbol{x}_i), & \text{hinge loss} \\ \operatorname{argmax}_{h(\cdot),r} \sum_{i=1}^{m}\left[\delta_{r,y_i}\left(\sum_l U_{il}\right) - U_{ir}\right]h(\boldsymbol{x}_i), & \text{logistic} \end{cases}$$

  $\forall r, \forall h(\cdot) \in \mathcal{H}$;
  2) If the stopping criterion has been met, we exit the loop.
  if $\left\|\sum_{i=1}^{m}\left[\delta_{r,y_i} - U_{ir}\right]h(\boldsymbol{x}_i)\right\|_2 < \nu + \epsilon$ **then**
    break; (hinge loss)
  if $\left\|\sum_{i=1}^{m}\left[\delta_{r,y_i}\left(\sum_l U_{il}\right) - U_{ir}\right]h(\boldsymbol{x}_i)\right\|_2 < \nu + \epsilon$ **then**
    break; (logistic loss)
  3) Add the best weak learner, $h_t(\cdot)$, into the current set;
  4) Solve either the primal or the dual problem (we solve the dual (5)) for the hinge loss case; or solve the primal problem (10) using ADMM for the logistic loss case;
  5) Update sample weights (dual variables);
  6) $t \leftarrow t + 1$;

The details of our boosting algorithm are given in Algorithm 1.

Implementation Note that the dual problem of hinge loss, (5), is a conic quadratic optimization problem involving several linear constraints and quadratic cones. We use the Mosek optimization solver to solve (5) which provides solutions for both primal and dual problems simultaneously using the interior-point method. For the logistic loss formulation the primal problem has nk variables and mk simple constraints (10). The dual problem has mk variables[2] and nk constraints. In boosting, we often have more training samples than final weak classifiers ($m \gg n$). However, the $\ell_{1,2}$-norm is not differentiable everywhere, and thus to solve (10) we apply the ADMM method [4]. ADMM decouples the regularization term from the logistic loss by introducing additional auxiliary variables. The algorithm then solves (10) by using an alternating minimization approach. A brief summary of ADMM in provided in Algorithm 2. See the supplementary for details of using ADMM to solve our mixed-norm regularization problems.

2.1. Faster training of multi-class boosting

In the last section, although we have combined ADMM with L-BFGS-B for faster training of multi-class logistic loss, the resulting algorithm is still computationally expensive to train. The drawback of (10) is that the formulation cannot be separated for faster training. Since real-world

[2]Here we ignore the equality constraints since they can be put back into the original cost function.

Algorithm 2 ADMM for solving (10)

Input:
1) Outputs of weak classifiers, H; 2) Augmented Lagrangian parameter, λ;
3) The maximum number of iterations, $s_{\max}$;
Output: An optimal W^*;
Initilaize: 1) $s \leftarrow 0$; 2) W^0, Z^0, U^0;

1 **repeat**
2 $\quad W^{s+1} = \underset{W}{\operatorname{argmin}} \frac{1}{mk} \sum_{i=1}^{m} \sum_{r=1}^{k} \log\big(1+\exp\left(-\rho_{ir}\right)\big) +$
3 $\quad\quad (U^s)^\top W + \frac{\lambda}{2}\|W - Z^s\|_2^2$;
4 $\quad Z_{j:}^{s+1} = \mathcal{S}_{\lambda/\rho}(W_{j:}^{s+1} + U_{j:}^s), \forall j$
5 $\quad$ where $\mathcal{S}_\kappa(a) = (1-\kappa/\|a\|_2)_+ a$;
6 $\quad U^{s+1} = U^s + \lambda(W^{s+1} - Z^{s+1})$;
7 $\quad s \leftarrow s+1$;
8 $\quad$ **if** $s > s_{\max}$ **then**
9 $\quad\quad$ break;
10 **until** convergence ;

data often consists of a large number of samples and classes, the training procedure can be very slow.

In order to improve the training efficiency of the classifier we thus propose here another variation of the multi-class boosting based on the logistic loss. This variation is achieved through a simplification of the form of ρ_{ir} in (10) to $\rho_{ir} = y_{ir} H_{i:} \boldsymbol{w}_r$ where $y_{ir} = 1$ if $y_i = r$ and $y_{ir} = -1$, otherwise. Note that this formulation was originally introduced in [7] for multi-class as well as multi-label support vector machine (SVM) learning and proved to be effective. To our knowledge, this formulation of multi-class loss function has not been applied to boosting. Here we extend it to multi-class boosting. The fast training (FAST) formulation is:

$$\min_{W,\boldsymbol{\rho}} \frac{1}{mk} \sum_{i=1}^{m} \sum_{r=1}^{k} \log\big(1+\exp\left(-\rho_{ir}\right)\big) + \nu\|W\|_{1,2} \quad (14)$$
$$\text{s.t. } \rho_{ir} = y_{ir} H_{i:} \boldsymbol{w}_r, \forall i, \forall r;\ W \geq 0.$$

The Lagrange dual can be written as

$$\max_{U} \ -\frac{1}{mk} \sum_{i=1}^{m} \sum_{r=1}^{k} \Big[mkU_{ir} \log\left(mkU_{ir}\right) + \quad (15)$$
$$\left(1 - mkU_{ir}\right) \log\left(1 - mkU_{ir}\right) \Big]$$
$$\text{s.t.} \quad \textstyle\sum_i U_{ir} y_{ir} H_{i:} \leq \nu Q_{:r}, \forall r;\ \|Q_{j:}\|_2 \leq 1, \forall j.$$

The relationship between ρ and U_{ir} is the same as (12). We replace steps 1 and 2 in Algorithm 1 with the constraint in (15) and step 4 in Algorithm 1 with the optimization problem in (14). As in [7], it is easy to apply the above formulation to multi-label classification, where each example can have multiple class labels. We leave this for future work.

Parallel optimization for FAST boosting The bottleneck of Algorithm 2 lies in minimizing W^{s+1}. By simplifying the margin as $\rho_{ir} = y_{ir} H_{i:} \boldsymbol{w}_r$, we can solve each $\boldsymbol{w}_r, \forall r$ independently. This speeds up our training time by a factor proportional to the number of classes. Let us define $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_k] \in \mathbb{R}^{n \times k}$, $Z = [\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_k] \in \mathbb{R}^{n \times k}$ and $U = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_k] \in \mathbb{R}^{n \times k}$, line 2 in Algorithm 2 can simply be replaced by,

$$\boldsymbol{w}_r^{s+1} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{mk} \sum_{i=1}^{m} \log\big(1+\exp\left(-\rho_{ir}\right)\big) + \quad (16)$$
$$(\boldsymbol{u}_r^s)^\top \boldsymbol{w} + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{z}_r^s\|_2^2, \quad \forall r.$$

Even without a multi-core processor, solving a series of (16) is still faster than solving line 2 in Algorithm 2. Distributed optimization can also be applied to our algorithms to further speed up the training time. The idea is to distribute a subset of training data in (16) to each processor and gather optimal $\boldsymbol{w}_r^{s+1}$ to form the average. Interested readers should refer to Chapter 8 in [4].

3. Experiments

In order to ensure a fair comparison we evaluate the performance of the proposed algorithms against other multi-class algorithms using binary weak learners: AdaBoost.MH [21], AdaBoost.ECC [16], MultiBoost$^{\ell_1}$ [23], AdaBoost-SIP [29], JointBoost [26], GradBoost (ℓ_1/ℓ_2-regularized) [11]. Note that the last three also try to share features across classes. For AdaBoost.ECC, we use the random-half partitioning technique, where we randomly assign half of the classes to be positive [18]. Decision stumps are chosen as the weak classifier for all boosting algorithms due to their simplicity and efficiency. For our algorithm, we mainly use the $\ell_{1,2}$ regularization since $\ell_{1,\infty}$ delivers similar performance.

Artificial data We consider the problem of discriminating 6 object classes on a 2D plane. Each sample consists of 2 measurements: orientation and radius. For all classes, the orientation is drawn uniformly between 0 and 2π. The radius of the first group is drawn uniformly between 0 and 1, the radius of the second group between 1 and 2, and so on. We generate 50 samples in the first group, 100 samples in the second group, 150 samples in the third group, and so on. The number of training sets is the same as the number of test sets. In this example feature vectors are the vertical and horizontal coordinates of the samples. We train 5 different classifiers based on the proposed MultiBoost$^{\text{group}}$ (hinge loss), AdaBoost.MH [21], AdaBoost.ECC [16], JointBoost [26] and MultiBoost$^{\ell_1}$ [23]. The multi-class classifier is composed of a set of binary decision stumps. For our algorithm and MultiBoost$^{\ell_1}$, we choose the regularization parameter ν from $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. For JointBoost, we set the outermost class (maximal radius) as background. We evaluate 5 boosting algorithms on this toy data and plot the decision boundary in Fig. 1. Table 1 reports some training and test error rates. Our algorithm performs best amongst five evaluated classifiers. We conjecture that the poor performance of JointBoost is due to the small number of background samples in the training data. JointBoost was designed for the task of multi-class object detection where

# feat.	Ada.ECC [16]	Ada.MH [21]	Joint [26]	Multi [23]	Ours
20	0.62/0.68	0.48/0.53	0.71/0.71	0.10/**0.14**	0.10/**0.14**
100	0.23/0.33	0.17/0.24	0.44/0.50	0.05/0.13	0.03/**0.10**
500	0.08/0.20	0.09/0.18	0.24/0.38	0.03/0.10	0.02/**0.09**

Table 1: Training/test errors of a few multi-class boosting methods on the 2D toy data set. The proposed MultiBoost$^{\text{group}}$ with hinge loss performs slightly better than others. See Fig. 1 for an illustration.

the objective is to detect several classes of objects from background samples. The algorithm might not work well on general multi-class problems. We then repeat our experiment by increasing the number of iterations to 500, and JointBoost, Adaboost.MH and AdaBoost.ECC still perform poorly on this toy data set compared to our approach.

UCI data sets The second experiment is carried out on some UCI machine learning data sets. Since we are more interested in the performance of multi-class algorithms when the number of classes is large, we evaluate our algorithm on 'segment' (7 classes), 'USPS' (10 classes), 'pendigits' (10 classes), 'vowel' (11 classes) and 'isolet' (26 classes). All data instances from 'segment' and 'vowel' are used in our experiment. For USPS, pendigits and isolet we randomly select 100 samples from each class. We use the original attributes for USPS (256 attributes) and isolet (617 attributes). For the rest, we increase the number of attributes by multiplying pairs of attributes. Each data set is then randomly split into two groups: 75% samples for training and 25% for evaluation. In this experiment, we compare MultiBoost$^{\text{group}}$ (logistic loss) to AdaBoost.MH [21], AdaBoost.ECC [16], GradBoost (ℓ_1/ℓ_2-regularized) [11] and MultiBoost$^{\ell_1}$ [23]. The regularization parameter is first determined by 5-fold cross validation.

For GradBoost, we choose the regularization parameter from $\{10^{-4}, 5{\cdot}10^{-4}, 10^{-3}, 5{\cdot}10^{-3}, 10^{-2}, 5{\cdot}10^{-2}, 10^{-1}, 5{\cdot}10^{-1}\}$. For MultiBoost$^{\ell_1}$ and our algorithm, we choose the regularization parameter from $\{10^{-7}, 5\cdot10^{-7}, 10^{-6}, 5\cdot10^{-6}, 10^{-5}, 5{\cdot}10^{-5}, 10^{-4}, 5{\cdot}10^{-4}, 10^{-3}\}$. All experiments are repeated 10 times using the same regularization parameter. The maximum number of boosting iterations is set to 500. We observe that almost all the algorithms converge earlier than 500 in this experiment. We plot the mean of test errors versus proportion of features used in Fig. 2. These results show that our proposed approach consistently outperforms its competitors. On the 'segment' and 'vowel' data sets we observe that our algorithm performs similarly to MultiBoost$^{\ell_1}$. We suspect that this is because the number of attributes in both data sets is quite small, and thus that there is little advantage to be gained through feature sharing on these data sets. Our approach often has the fastest convergence rate (note, however, that GradBoost converges faster on the USPS data sets but ends up with a larger test error).

Comparison between GradBoost and our algorithm GradBoost with mixed-norm regularization [11] is similar to the method presented here. The distinction, however, is that our method minimizes the original convex loss function rather than quadratic bounds on this function. The result is that our method is not only more effective, but also more general, as it can be applied not only to the logistic loss function but also to any convex loss function. In addition, our approach shares a similar formulation to standard boosting algorithms, *i.e.*, the way we generate weak learners or update sample weights (dual variables in our algorithm). The algorithm of [11] is rather heuristic and it is not known when the algorithm will converge. Furthermore, GradBoost is more similar to FloatBoost [19] where the authors introduce a backward pruning step to remove less discriminative weak classifiers. The drawback of pruning is 1) being heuristic and 2) a prolonged training process.

	MNIST	ABCDETC
Ada.MH [21]	**3.0** (0.2)	63.4 (1.8)
Ada.ECC [16]	3.1 (0.2)	70.5 (1.1)
Ada.SIP [29]	4.4 (1.3)	62.7 (1.2)
GradBoost [11]	5.3 (0.3)	73.9 (1.3)
MultiBoost [23]	3.7 (0.2)	73.2 (0.7)
MultiBoost$^{\text{group}}$(ours)	3.1 (0.2)	59.1 (1.1)
MultiBoost$^{\text{group}}_{\text{FAST}}$(ours)	**3.0** (0.3)	**58.2** (0.9)

Table 2: Test errors (%) of a few multi-class boosting methods on the MNIST and ABCDETC handwritten data sets. All experiments are run 10 times with 500 boosting iterations. The average error mean and standard deviation (in percentage) are reported.

ABCDETC and MNIST handwritten data The NEC Lab ABCDETC sets consist of 72 classes (digits, letters and symbols). For this experiment, we only use digits and letters (10 digits, 26 lower cases and 26 upper cases). We first resize the original images to a resolution of 28×28 pixels and apply a de-skew pre-processing. We then apply a spatial pyramid and extract 3 levels of HOG features with 50% block overlap. The block size in each level is 4×4, 7×7 and 14×14 pixels, respectively. Extracted HOG features from all levels are concatenated. In total, there are 2,172 HOG features. For ABCDETC, we randomly select 5 samples from each class as training sets and 120 samples from each class as test sets. For MNIST, we randomly select 100 samples from each class as training sets and used the original test sets of 10,000 samples. In this experiment, we also compare the performance of MultiBoost$^{\text{group}}$ with a fast training variant, MultiBoost$^{\text{group}}_{\text{FAST}}$. All experiments are run 10 times with 500 boosting iterations and the results are briefly summarized in Table 2. From the table, both MultiBoost$^{\text{group}}$ and MultiBoost$^{\text{group}}_{\text{FAST}}$ perform best compared to other evaluated algorithms, especially on ABCDETC test sets where the number of classes is large. We observe the FAST approach to perform slightly better than MultiBoost$^{\text{group}}$. In our work, the advantage of the FAST approach compared to MultiBoost$^{\text{group}}$ is that the training time can be further reduced by exploiting parallelism in ADMM, as previously mentioned. Table 3 illustrates the feature sharing property of our algorithms. Clearly we can see that the group sparsity regularization indeed encourages sharing features.

Scene recognition In the next experiment, we compare

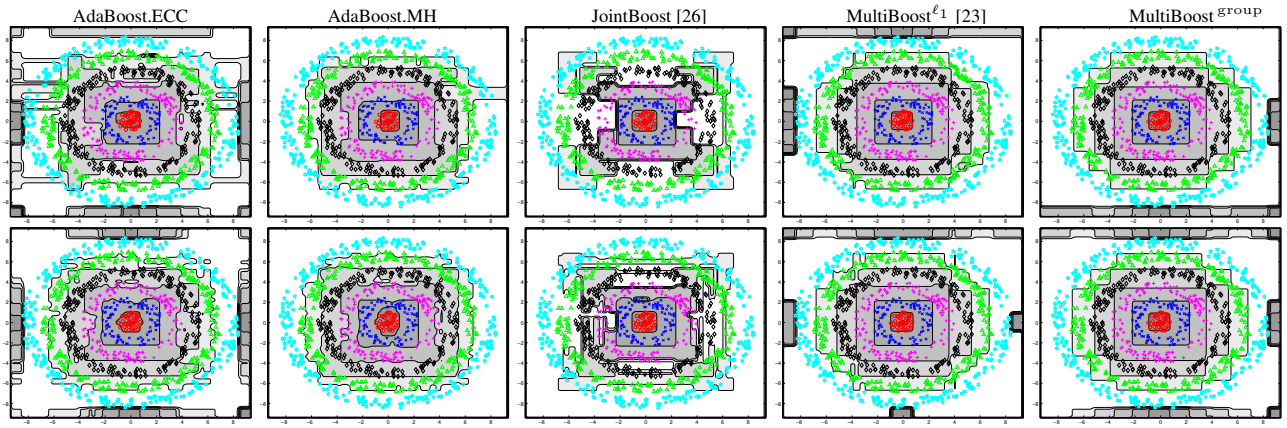

Figure 1: Decision boundaries on a toy data sets, with **Top row:** 100 weak classifiers and **Bottom row:** 500 weak classifiers. Note that some multi-class algorithms end up with very complicated and multi-modal decision boundaries.

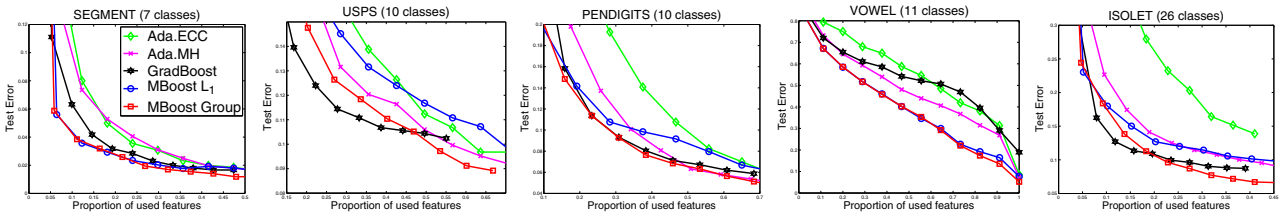

Figure 2: The performance of our algorithm (MultiBoost$^{\text{group}}$) compared with various boosting algorithms on several machine learning data sets. The horizontal axis is the fraction of used features and the vertical axis is the test error rate. We observe that group sparsity-based approaches (ours and GradBoost) generally converge faster than other algorithms.

MNIST	'0 − 3'	'4 − 5'	'6 − 7'	'8 − 10'
MultiBoost$^{\ell_1}$	99.8%	0.2%	0%	0%
MultiBoost$^{\text{group}}$	4.5%	48.8%	40.9%	5.8%
MultiBoost$^{\text{group}}_{\text{FAST}}$	10.1%	69.9%	19.7%	0.3%
ABCDETC	'0 − 15'	'16 − 30'	'31 − 45'	'46 − 62'
MultiBoost$^{\ell_1}$	99.8%	0.2%	0%	0%
MultiBoost$^{\text{group}}$	0%	81.3%	18.7%	0%
MultiBoost$^{\text{group}}_{\text{FAST}}$	0%	65.7%	33.5%	0.7%

Table 3: The distribution of shared weak classifiers. For example, '8 − 10' indicates that the weak classifier is being shared among 8 to 10 classes. The table illustrates the feature sharing property of our algorithms, *i.e.*, one weak classifier is being shared among multiple classes.

our approach on the 15-scene data set used in [17]. The set consists of 9 outdoor scenes and 6 indoor scenes. There are 4,485 images in total. For each run, the available data are randomly split into a training set and a test set based on published protocols. This is repeated 5 times and the average accuracy is reported. In each train/test split, a visual codebook is generated using only training images. Both training and test images are then transformed into histograms of code words. We use CENTRIST [27] as our feature descriptors. 200 visual code words are built using the histogram intersection kernel (HIK), which has been shown to outperform k-means and k-median [27]. We represent each image in a spatial hierarchy manner [3]. Each image consists of 31 sub-windows. An image is represented by the concatenation of histograms of code words from all 31 sub-windows. Hence, in total there are 6,200 dimensional histogram.

Fig. 3 shows the average classification errors. We observe that both MultiBoost$^{\text{group}}$ and MultiBoost$^{\ell_1}$ [23] converge quickly in the beginning. However, our approach has a better overall convergence rate. We also observe that both of our approaches, (MultiBoost$^{\text{group}}$ and MultiBoost$^{\text{group}}_{\text{FAST}}$), have the lowest test error compared to other algorithms evaluated. We also apply a multi-class SVM to the above data set using the LIBSVM package [6] and report the recognition results in Table 4. SVM with 6,200 features achieves an average accuracy of 76.30% (linear) and 81.47% (non-linear). Our results indicate that both proposed approaches achieve a comparable accuracy to non-linear SVM while requiring less number of features (77.8% accuracy for MultiBoost$^{\text{group}}$ with 1000 features and 79.2% accuracy for MultiBoost$^{\text{group}}_{\text{FAST}}$).

methods	# features used	accuracy (%)
SAMME† [30]	1000	70.9 (0.40)
JointBoost† [26]	1000	72.2 (0.70)
MultiBoost$^{\ell_1}$ [23]	1000	76.0 (0.48)
AdaBoost.SIP [29]	1000	75.7 (0.10)
AdaBoost.ECC [16]	1000	76.5 (0.67)
AdaBoost.MH [21]	1000	77.6 (0.59)
MultiBoost$^{\text{group}}$ (ours)	1000	77.8 (0.77)
MultiBoost$^{\text{group}}_{\text{FAST}}$(ours)	**1000**	**79.2** (0.82)
Linear SVM	6200	76.3 (0.88)
Nonlinear SVM (HIK)	**6200**	**81.4** (0.60)

Table 4: Recognition rate of various algorithms on Scene15 data sets. All experiments are run 5 times. The average accuracy mean and standard deviation (in percentage) are reported. Results marked by † were reported in [29].

Traffic sign recognition We evaluate our approach on the recent German traffic sign recognition benchmark[3].

[3] http://benchmark.ini.rub.de/

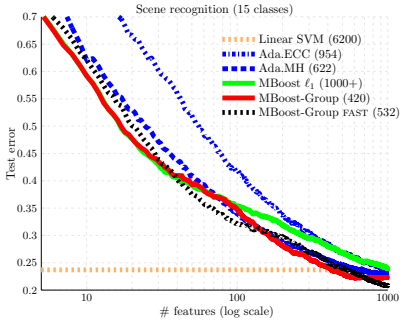

Figure 3: Performance of different classifiers on the scene recognition data set. We also report the number of features required to achieve similar results to linear multi-class SVM. Both of our methods (MultiBoost$^{\text{group}}$ and MultiBoost$^{\text{group}}_{\text{FAST}}$) outperform other evaluated boosting algorithms.

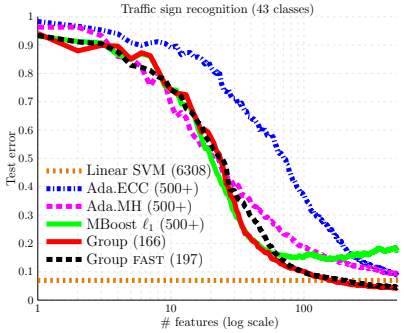

Figure 4: Performance of different classifiers on traffic sign recognition data sets. We also report the number of features needed to achieve a similar accuracy to the linear SVM. Both of our methods outperform other multi-class methods in terms of the test error.

Data sets consist of 43 classes with more than $50,000$ images in total. We randomly select 100 samples from each class to train our classifier. We use the provided test set to evaluate the performance of our classifiers ($12,569$ images). All training images are scaled to 40×40 pixels using bilinear interpolation. Three different types of pre-computed HOG features are provided ($6,052$ features). We combine all three types together. We also make use of histogram of hue values (256 bins). Hence, there is a total of $6,308$ features. The results of different classifiers are shown in Fig. 4. Our proposed classifier outperforms other evaluated classifiers. As a baseline, we train a multi-class SVM using LIBSVM [6]. SVM achieves 93.05% (using $6,308$ features) while our classifier achieves 95.62% for MultiBoost$^{\text{group}}$ and 95.42% for MultiBoost$^{\text{group}}_{\text{FAST}}$ with a much smaller set of features (500 features). Note that an overfitting behavior is observed for MultiBoost$^{\ell_1}$.

4. Conclusion

We have proposed a new feature-sharing multi-class boosting method. The proposed boosting is based on the primal-dual view of the group sparsity regularized optimization. We derive the Lagrange dual problems and using column generation to implement the totally corrective boosting. Extensive experiments show the excellence of the proposed algorithm. We plan to extend our framework to a hierarchical classification model where objects in the same category, *e.g.*, buses and trucks, can share visual appearance. We will also explore the possibility of applying the proposed framework to real-time object detection [20, 25].

Acknowledgement This work is supported in part by the Australian Research Council Linkage Project LP100100791.

References

[1] The mosek optimization software, version 6.0, 2011. `http://www.mosek.com`.

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

[3] A. Bosch, A. Zisserman, and X. Munoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(4):712 – 727, 2008.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations & Trends in Mach. Learn.*, 3(1), 2011.

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

[6] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Sys. & Tech.*, 2(3), 2011.

[7] O. Chapelle and S. S. Keerthi. Multi-class feature selection with support vector machines. In *Proc. American Stat. Assoc.*, 2008.

[8] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector mchines. *J. Mach. Learn. Res.*, 2:265–292, 2001.

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

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

[11] J. Duchi and Y. Singer. Boosting with structural sparsity. In *Proc. Int. Conf. Mach. Learn.*, 2009.

[12] Y. Freund. An adaptive version of the boost by majority algorithm. *Mach. Learn.*, 43(3):293–318, 2004.

[13] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. Int. Conf. Mach. Learn.*, 1996.

[14] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comp. & Sys. Sciences*, 55:119–139, 1997.

[15] J. Fürnkranz. Round robin classification. *J. Mach. Learn. Res.*, 2:721–747, 2002.

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

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2006.

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

[19] S. Z. Li and Z. Zhang. FloatBoost learning and statistical face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1112–1123, 2004.

[20] S. Paisitkriangkrai, C. Shen, and J. Zhang. Fast pedestrian detection using a cascade of boosted covariance features. *IEEE Trans. Circuits Syst. Video Technol.*, 18(8):1140–1151, 2008.

[21] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated prediction. *Mach. Learn.*, 37(3):297–336, 1999.

[22] R. E. Schapire. Using output codes to boost multiclass learning problems. In *Proc. Int. Conf. Mach. Learn.*, 1997.

[23] C. Shen and Z. Hao. A direct formulation for totally-corrective multi-class boosting. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.

[24] C. Shen and H. Li. On the dual formulation of boosting algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010.

[25] C. Shen, S. Paisitkriangkrai, and J. Zhang. Face detection from few training examples. In *Proc. IEEE Int. Conf. Image Process.*, 2008.

[26] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(5):854–869, 2007.

[27] J. Wu and J. M. Rehg. CENTRIST: A visual descriptor for scene categorization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1489–1501, 2011.

[28] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.*, 5:975 – 1005, 2004.

[29] B. Zhang, G. Ye, Y. Wang, J. Xu, and G. Herman. Finding shareable informative patterns and optimal coding matrix for multiclass boosting. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2009.

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