RECYCLED LINEAR CLASSIFIERS FOR MULTICLASS CLASSIFICATION

Akshay Soni, Jarvis Haupt

Department of Electrical and Computer Engineering
University of Minnesota, Twin Cities
Minneapolis, MN 55414
{sonix022, jdhaupt}@umn.edu

Fatih Porikli

Australian National University
Canberra, ACT, Australia
fatihporikli@ieee.org

ABSTRACT

Many machine learning applications employ a multiclass classification stage that uses multiple binary linear classifiers as building blocks. Among these, commonly used strategies such as one-vs-one classification can require learning a large number of hyperplanes, even when the number of classes to be discriminated among is modest. Further, when the data being classified is inherently high-dimensional, the storage and computational complexity associated with the application of multiple linear classifiers can ignite critical resource management issues. This work describes a novel multiclass classification method based on efficient use of a single "recycled" linear classifier (or ReLiC), which addresses these storage and implementation complexity issues. The proposed approach amounts to constraining the entire collection of hyperplanes to be circularly-shifted versions of each other, enabling classification procedures that may be implemented with efficient operations, such as circular convolution (which can be efficiently computed using transform domain techniques), and simple sampling/thresholding operations. We show that the optimization task associated with our proposed approach can be formulated as a quadratic program, and we introduce an efficient distributed procedure for its solution based on an alternating direction method of multipliers. Simulation results demonstrate that the performance of the proposed approach is comparable with the more complex, traditional multiclass linear classification strategies, suggesting the proposed approach is a viable alternative in large-scale data classification tasks.

1. INTRODUCTION

We consider the problem of multiclass classification in a high-dimensional Euclidean space within a supervised learning framework. Given a training data set of the form $(\mathbf{x}_i, \ell_i)$, where $\mathbf{x}_i \in \mathbb{R}^n$ is the ith example and $\ell_i \in \{1, \ldots, K\}$ is the ith class label, supervised multiclass classification aims to assign a class label to new examples. Multiple approaches have been proposed to address this problem, some of which are naturally suited to multiclass classification (e.g., k-Nearest Neighbor [1], Naive Bayes classifiers [2]), and others that correspond to extensions of binary classification strategies (e.g., decision trees [3], neural networks [4], and Support Vector Machines (SVMs) [5]).

A commonly used approach along these lines is to reduce the overall multiclass classification task to K binary problems, where each problem discriminates a given class from the other $K-1$ classes. This one-versus-all (OvA) approach [6] requires a total of K binary classifiers, where the k-th classifier is trained with positive examples belonging to class k and negative examples belonging to the other $K-1$ classes. Another widely used approach is the one-versus-one (OvO) strategy [7], which aims to learn binary classifiers to discriminate between each pair of classes, requiring a total of $K(K-1)/2 \sim K^2$ unique classifiers. The quadratic dependence on the number of classes implies that learning and testing in this case can be very slow, especially when the number of classes is large and/or the data are high-dimensional.

Hierarchical Classifiers (HC) are also commonly used for multiclass classification tasks. Binary Hierarchical Classifiers [8], for example, learn and employ a total of $K-1$ binary classifiers, which are arranged in a binary tree with K leaf nodes, each corresponding to one class. Classification in these settings is performed in a top-down fashion: classification of a new pattern is achieved by traversing a path from the root node to a leaf node, where the path is determined by the outcome of the binary classifier at each node. Hierarchical SVM [9] uses a similar approach where clustering is performed via arranging classes into an undirected graph with edge weights representing the Kullback-Leibler distances between the classes, and employs a max-cut algorithm to split the classes into two sub-clusters that are most distant from each other. In either case, when the underlying tree is balanced (or nearly so), the classification task can be accomplished by implementing $O(\log K)$ binary tests.

Error correcting output coding (ECOC) incorporates the idea of codebooks of length N for each class according to a binary matrix [10], where each row corresponds to a certain class and each column is one of the N classifiers. When testing an unseen example, the output codeword from the N classifiers is compared to the given K codewords, and the one with the minimum hamming distance is considered the class label for that example. ECOC can be seen as a generalization of the OvO, OvA, and HC strategies. While the initial paper [10] discussed approaches using as many as $N \sim 2^K$ classifiers, variations of this approach have also been proposed which use many fewer ($N \sim \log K$) classifiers, see [11, 12]).

Overall for K classes, conventional OvA approaches require training of K unique binary classifiers, the OvO approach requires $K(K-1)/2$ binary classifiers, and binary tree classification requires learning $K-1$ binary classifiers. Similarly, ECOC-based schemes require, generally, learning some N classifiers where N may (in some instances) be large relative to K. For typical large-scale data classification tasks where number of classes K may also be large, these traditional approaches require learning, storing, and applying a large number of linear classifiers. In such settings, even the testing complexity can be prohibitive, requiring computation of multiple inner products between high-dimensional vectors.

A.S. and J. H. acknowledge support from DARPA/ONR Award No. N66001-11-1-4090.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

1.1. Our Contribution

In this work we propose a multiclass classification method based on efficient use of a single "**Re**cycled" **Li**near **C**lassifier (or **ReLiC**), which addresses the storage and implementation complexity issues associated with multiclass classification strategies that employ multiple binary linear classifiers. Our approach amounts to constraining the collection of hyperplanes to be circularly-shifted versions of each other, which facilitates classification procedures that may be implemented with efficient operations – circular convolution and simple sampling/thresholding operations.

The convolutional nature of our proposed approach is reminiscent of time-delay neural networks [13] and other convolutional neural networks [14], which employ convolutional weights in (multilayer, or deep belief) neural network architectures. Convolutional neural networks rely on the existence of (temporally or spatially) local features that are discriminative for the classification task, and employ *shift-invariant* feature extracting filters at each layer. In a general sense the storage and implementation complexity benefits associated with our proposed approach could also be enjoyed by related techniques based, for example, on *single-layer* convolutional neural networks. That said, there are a few key distinctions between convolutional networks and our proposed ReLiC approach. First, convolutional neural networks generally utilize spatially local filters; in contrast, the weight structure associated with our learned hyperplane classifiers are generally global. Further, neural networks can be prone to overfitting, and the limited spatial extent of the learned filters in convolutional neural networks serves as a form of regularization for the network training task. Our proposed approach, in contrast, is based on a support vector machine philosophy, in which margin and slack constraints serve as regularizers to prevent overfitting. For a detailed overview of the other linear classification and multiclass approaches, we refer reader to [15].

1.2. Outline

The remainder of this paper is organized as follows. In Section 2 we describe our Recycled Linear Classifier (ReLiC) approach, and explain how the resulting optimization problem can be interpreted within an SVM framework. We also discuss a modest extension using slack variables to allow for some errors in the classifier training phase. We describe a computationally-efficient distributed training approach (d-ReLiC) in Section 3. We provide some empirical experimental results in Section 4, demonstrating (for one standard dataset) that our proposed approach sacrifices little accuracy relative to existing multi class SVM approaches. A few conclusions and extensions are discussed in Section 5.

2. RECYCLED LINEAR CLASSIFIERS (RELIC)

In this section we introduce the ReLiC approach, the key idea of which is to learn hyperplanes that are circularly shifted versions of each other. The implication of this design is that multiple individual classifiers can be applied to a new data point essentially in parallel, and in an efficient manner, using fast Fourier transform methods. In contrast, traditional multiclass classification strategies based on linear SVMs require computing projections of the test data to be classified onto multiple individual (and generally, structurally unrelated) hyperplanes, which can be significantly more costly in terms of storage and computational complexity.

This difference is highlighted in Table 1. The table depicts a comparison of the storage and implementation complexities of the OvO, OvA and HC classification approaches described above, and for the proposed ReLiC approach, for data of dimension n. The storage complexity of our proposed approach is uniformly more favorable than the analogous traditional approaches. Further, the implementation complexity of our approach may be significantly less (and is never worse) than that of the traditional approaches, though this improvement depends inherently on the number of classes being discriminated among.

Table 1: Storage and testing time requirements for different multiclass schemes and ReLiC in terms of the ambient data dimension n and number of classes K.

	Storage Complexity	Testing Complexity
OvA	$\mathcal{O}(nK)$	$\mathcal{O}(nK)$
OvO	$\mathcal{O}(nK^2)$	$\mathcal{O}(nK^2)$
HC	$\mathcal{O}(nK)$	$\mathcal{O}(nK)$
ReLiC (OvA)	$\mathcal{O}(n)$	$\mathcal{O}\left(n \cdot \min\{K, \log n\}\right)$
ReLiC (OvO)	$\mathcal{O}(n)$	$\mathcal{O}\left(n \cdot \min\{K^2, \log n\}\right)$
ReLiC (HC)	$\mathcal{O}(n)$	$\mathcal{O}\left(n \cdot \min\{K, \log n\}\right)$

Our learning formulation for ReLiC amounts to a constrained form of multiclass SVM based on binary classifiers. We briefly review some essential aspects of these approaches, in order to put our proposed formulation in context.

2.1. Multiclass Classification Using Multiple Hyperplanes

The training tasks associated with each of the traditional multiclass classification approaches based on multiple linear binary classifiers described above can be interpreted in terms of a unified framework. Suppose we partition the training data according to its label, so that the data belonging to class j comprise a set $\mathcal{X}_j$ for $j = 1, ..., K$. Suppose the overall strategy uses some N individual binary tests, then the aim of the training task amounts to identifying some N hyperplanes, denoted $\mathbf{h}_1, \ldots, \mathbf{h}_N$, and associated scalar biases, denoted $b_1, \ldots, b_N$, chosen so that the classification error using this collection of binary classifiers is small (or zero) over the training data set.

Depending on the strategy employed, only a subset of the N binary tests may take part in determining the label for a particular class. For example, determining whether a test vector belongs to class j using a OvO strategy depends only on the outcomes of the tests comparing j with each of the other classes, while tests comparing any other pair of classes (say p and q with $p, q \neq j$) are not used when determining membership in class j. Likewise, classification strategies based on hierarchical trees employ only a subset of the binary tests, corresponding to the path through the tree identified by the outcomes of the previous tests. Formally, we prescribe to each class j a unique vector $\mathbf{y}^{(j)} \in \{-1,\ 0,\ 1\}^N$. If for $\mathbf{x} \in \mathcal{X}_j$ we denote $z_i = \mathbf{h}_i^{\mathrm{T}}\mathbf{x} + b_i$, and let $\mathbf{z} = [z_1, ..., z_N]^{\mathrm{T}}$, then enforcing that $\mathbf{x}$ belong to class j amounts to ensuring that $\mathrm{diag}(\mathbf{y}^{(j)})\mathbf{z} \geq \mathbf{0}$, where $\mathbf{y}^{(j)} = [y_1^{(j)}, ..., y_N^{(j)}]^{\mathrm{T}}$ is a pre-determined label vector associated with the j-th class, and the operator $\geq$ applies componentwise. Here, setting $y_i^{(j)} = 0, i = 1, ..., N$ enforces that the ith hyperplane $\mathbf{h}_i$ does not take part in assessing whether $\mathbf{x}$ belongs to class j – this amounts to a kind of "don't care" condition for that test.

In our formulation (presented in the next sub-section) we will find it advantageous to work with a slightly modified condition. For each class j we can define a non-zero row selection matrix $\mathbf{Q}^{(j)}$ that corresponds to a row sub matrix of the identity matrix, and whose nonzero rows select the nonzero elements of the class label vector $\mathbf{y}^{(j)}$. With this, we can express the condition for membership in class j as $\mathbf{Q}^{(j)}\mathrm{diag}(\mathbf{y}^{(j)})\mathbf{z} \geq \mathbf{1}$. Note that choice of $\mathbf{1}$ is arbitrary,

since we can always scale the elements of the learned hyperplanes and scalar offsets to scale the magnitude of $\mathbf{z}$. Linear inequalities of this form (or relaxed versions, when training errors are allowed), one per data point, arise as constraints in the overall training problem.

2.2. "Recycling" Linear Classifiers

The essential idea behind ReLiC is that each hyperplane $\mathbf{h}_i$ should be constrained to be a circularly shifted version of some base hyperplane $\mathbf{h}$. In this case we can express $\mathbf{z}$ more compactly as $\mathbf{z} = \mathbf{SHx} + \mathbf{b}$, where $\mathbf{S} \in \mathbb{R}^{N\times n}$ is a fixed selection matrix corresponding to a row submatrix of the identity matrix having N rows, $\mathbf{H} \in \mathbb{R}^{n\times n}$ is a circulant matrix whose first row is $\mathbf{h}$, and $\mathbf{b} = [b_1, ..., b_N]^{\mathrm{T}}$ is a vector of scalars. The operation of multiplying of a circulant matrix with a vector is equivalent to circular convolution of the first row of the circulant matrix with the vector; in light of this, we may express the condition that $\mathbf{x}$ belong to class j as

$$\begin{aligned}\mathbf{Q}^{(j)}\mathrm{diag}(\mathbf{y}^{(j)})\mathbf{z} &= \mathbf{Q}^{(j)}\mathrm{diag}(\mathbf{y}^{(j)})(\mathbf{SHx}+\mathbf{b}),\\ &= \mathbf{Q}^{(j)}\mathrm{diag}(\mathbf{y}^{(j)})(\mathbf{S}(\mathbf{h}\circledast\mathbf{x})+\mathbf{b}) \geq \mathbf{1},\end{aligned}$$

where $\circledast$ denotes the circular convolution operator and $\mathbf{Q}^{(j)}$ is as defined in last subsection.

An important property of circulant matrices, which will be key ingredient of our formulation, is that any circulant matrix $\mathbf{H}$ can be written as

$$\mathbf{H} = \sum_{i=1}^{n} h_i \mathbf{P}^{i-1}, \tag{1}$$

where $h_i,\ i = 1, ..., n$ are scalars and $\mathbf{P} \in \mathbb{R}^{n\times n}$ is the cyclic permutation matrix given as

$$\mathbf{P} = \begin{bmatrix} \mathbf{0}_{n-1}^{\mathrm{T}} & 1 \\ \mathbf{I}_{n-1} & \mathbf{0}_{n-1} \end{bmatrix}, \tag{2}$$

where $\mathbf{I}_{n-1}$ denotes the $(n-1)\times(n-1)$ identity matrix and $\mathbf{0}_{n-1}$ is an $(n-1)\times 1$ vector of zeros. Note that $\mathbf{P}^0 = \mathbf{I}_n$. For shorthand we write $\mathbf{H} = \mathrm{circ}(\mathbf{h})$, where $\mathbf{h} = [h_1\ h_2\ \ldots h_n]^{\mathrm{T}}$ is the vector of parameters associated with the representation (1).

As with traditional SVM formulations our goal is to design a set of linear classifiers having maximum margins $2/\|\mathbf{h}_i\|_2$ for $i = 1, \ldots, N$. Here, the margins for all the classifiers is same for our case since the classifiers are just linearly shifted versions of each other with same norm. With this observation, we may express our overall learning task as follows. Suppose that we have a total of T training data points $(\mathbf{x}_i, \ell_i)$ for $i = 1, \ldots, T$. Then, our aim is to solve

$$\begin{aligned}\underset{\mathbf{h},\,\mathbf{b}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{h}\|_2^2 \\ \text{s.t.} \quad & \mathbf{Q}^{(\ell_i)}\mathrm{diag}(\mathbf{y}^{(\ell_i)})(\mathbf{SHx}_i+\mathbf{b}) \geq \mathbf{1},\ i = 1, ..., T, \\ & \mathbf{H} = \mathrm{circ}(\mathbf{h}),\end{aligned} \tag{3}$$

where as described above, $\mathbf{y}^{(\ell_i)}$ denotes the (pre-determined) label vector associated with the i-th training data point where $\ell_i \in \{1, 2, \ldots, K\}$. We can further remove the circulant constraint from (3) and substitute directly $\mathbf{H} = \sum_{j=1}^{n} h_j \mathbf{P}^{(j-1)}$ by using the property (1) and rewrite the constraints in a more compact form by making the substitution $\sum_{j=1}^{n} h_j \mathbf{Q}^{(\ell_i)}\mathrm{diag}(\mathbf{y}^{(\ell_i)})\mathbf{S}\mathbf{P}^{j-1}\mathbf{x}_i + \mathbf{Q}^{(\ell_i)}\mathrm{diag}(\mathbf{y}^{(\ell_i)})\mathbf{b} = \mathbf{B}^i\mathbf{h} + \mathbf{C}^i\mathbf{b}$,, where $\mathbf{B}^i,\ i = 1, ..., T$ are matrices whose columns are the vectors $\mathbf{Q}^{(\ell_i)}\mathrm{diag}(\mathbf{y}^{(\ell_i)})\mathbf{S}\mathbf{P}^{j-1}\mathbf{x}_i,\ j = 1, ..., n$ and $\mathbf{C}^i = \mathbf{Q}^{(\ell_i)}\mathrm{diag}(\mathbf{y}^{(\ell_i)})$. The resulting optimization problem in $\mathbf{h}$ and $\mathbf{b}$ amounts to a quadratic program (QP) of the form

$$\begin{aligned}\underset{\mathbf{h},\mathbf{b}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{h}\|_2^2 \\ \text{s.t.} \quad & \mathbf{B}^i\mathbf{h} + \mathbf{C}^i\mathbf{b} \geq \mathbf{1},\ \ i = 1, ..., T.\end{aligned} \tag{4}$$

Note that since the objective function of (4) is strictly convex in $\mathbf{h}$ and the constraints are affine, the optimization problem is convex with unique global optimal solution $\mathbf{h}^*$ and $\mathbf{b}^*$. It is interesting to note that problem (4) has a form that is reminiscent of the optimization problem corresponding to linear SVM [16], which also may be cast as a QP.

As in the case of relaxed margin SVM, ReLiC can also be modified to allow some errors at the training stage. The optimization problem in this case becomes

$$\begin{aligned}\underset{\mathbf{h},\mathbf{b},\boldsymbol{\xi}^i}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{h}\|_2^2 + \eta\sum_{i=1}^{T}\|\boldsymbol{\xi}^i\|_1 \\ \text{s.t.} \quad & \mathbf{B}^i\mathbf{h} + \mathbf{C}^i\mathbf{b} \geq \mathbf{1} - \boldsymbol{\xi}^i \text{ and } \boldsymbol{\xi}^i \geq \mathbf{0},\ \ i = 1, ..., T,\end{aligned} \tag{5}$$

where $\boldsymbol{\xi}^i = [\xi_1^i, ...\xi_{k_i}^i]$ is a nonnegative vector of *slack* parameters, and $\eta \geq 0$ is a regularization parameter that trades off the classifier margin with the error allowed during training. Once we learn $\mathbf{h}$ and $\mathbf{b}$ by solving (4) (or its relaxed margin version (5)), the class assignment of an unseen sample $\mathbf{x}_{\mathrm{test}}$ is determined via $\mathbf{z} = \mathbf{S}(\mathbf{x}_{\mathrm{test}} \circledast \mathbf{h}) + \mathbf{b}$ by performing circular convolution of $\mathbf{x}_{\mathrm{test}}$ with $\mathbf{h}$ and adding the bias vector $\mathbf{b}$. The label decision is made according to the sign of the maximum entry of $\mathbf{z}$ for OvA, or by majority voting in the case of OvO.

A primary motivation of our approach is its applicability in large-scale data settings. In these cases the training phase may also be computationally demanding. In the next section we describe an efficient *distributed* approach for problems of the form (5) based on Alternating Direction Method of Multipliers (ADMM) [17].

3. DISTRIBUTED LEARNING FOR RELIC

We propose an ADMM based first order method to solve optimizations of the form (4) and (5), which facilitates an efficient distributed implementation across multiple machines or processing centers. In our approach, the training data is partitioned and distributed across multiple machines, so that each machine needs to work with only a small portion of training data and without any knowledge about the data present on the other machines. Our approach is a multi-step process. At each step, each machine performs its own local computations (described below) and sends a summary of its current estimate to a central machine where they are combined to form a global estimate for that step. The global estimates are then broadcast to each machine, and each uses this global information to update their own local estimates. In this way after few iterations the local and global variables achieve consensus and we can use the global variables as our final estimates. We refer to this distributed approach for learning recycled linear classifiers as *d-ReLiC*.

Our d-ReLiC procedure is obtained as follows. First, we represent (5) as a square loss optimization problem of the form (6) where η and c_1 are hyperparameters, and $\mathbf{s}^i$ are the non-negative slack vectors which converts the inequality constraints of (5) into equality constraints. Note that it is possible to use a hinge-loss term rather than square-loss, but the square-loss term leads to simple update steps. Our approach is motivated by prior works employing square-loss for least-squares based SVM [18, 19, 20].

$$\underset{\boldsymbol{\omega},\,\boldsymbol{\beta},\,\mathbf{s}^i\geq\mathbf{0},\,\boldsymbol{\xi}^i\geq\mathbf{0}}{\text{minimize}}\ \frac{1}{2}\|\boldsymbol{\omega}\|_2^2+\eta\sum_{i=1}^{T}\|\boldsymbol{\xi}^i\|_1+c_1\sum_{i=1}^{T}\|\mathbf{B}^i\boldsymbol{\omega}+\mathbf{C}^i\boldsymbol{\beta}-\mathbf{s}^i-\mathbf{1}+\boldsymbol{\xi}^i\|_2^2,\quad \text{s.t. } \boldsymbol{\omega}^*=\mathbf{h}^*,\boldsymbol{\beta}^*=\mathbf{b}^*, \tag{6}$$

$$\begin{aligned}\underset{\substack{\mathbf{h},\,\mathbf{b},\,\boldsymbol{\omega}_m,\,\boldsymbol{\beta}_m,\\ \mathbf{s}_m^i\geq\mathbf{0},\ \boldsymbol{\xi}_m^i\geq\mathbf{0}}}{\text{minimize}}\quad & \frac{1}{2}\|\mathbf{h}\|_2^2+\sum_{m=1}^{M}\left(\frac{\rho_1}{2}\|\boldsymbol{\omega}_m-\mathbf{h}\|_2^2+\frac{\rho_2}{2}\|\boldsymbol{\beta}_m-\mathbf{b}\|_2^2\right)+\sum_{i\in D_m}\left(c_1\|\mathbf{B}^i\boldsymbol{\omega}_m+\mathbf{C}^i\boldsymbol{\beta}_m-\mathbf{s}_m^i-\mathbf{1}+\boldsymbol{\xi}_m^i\|_2^2+\eta\|\boldsymbol{\xi}_m^i\|_1\right)\\ \text{s.t.}\quad & \boldsymbol{\omega}_m-\mathbf{h}=\mathbf{0},\ \boldsymbol{\beta}_m-\mathbf{b}=\mathbf{0},\ m=1,...,M,\end{aligned} \tag{7}$$

$$\begin{aligned}\underset{\substack{\mathbf{h},\mathbf{b},\,\boldsymbol{\omega}_m,\,\boldsymbol{\beta}_m,\\ \mathbf{s}_m^i\geq\mathbf{0},\,\boldsymbol{\xi}_m^i\geq\mathbf{0},\,\lambda,\,\mu}}{\text{mininimize}}\quad & \frac{1}{2}\|\mathbf{h}\|_2^2+\sum_{m=1}^{M}\sum_{i\in D_m}\left(c_1\|\mathbf{B}^i\boldsymbol{\omega}_m+\mathbf{C}^i\boldsymbol{\beta}_m-\mathbf{s}_m^i-\mathbf{1}+\boldsymbol{\xi}_m^i\|_2^2+\eta\|\boldsymbol{\xi}_m^i\|_1\right)\\ & +\sum_{m=1}^{M}\left(\frac{\rho_1}{2}\|\boldsymbol{\omega}_m-\mathbf{h}\|_2^2+\boldsymbol{\lambda}_m^{\mathrm{T}}(\boldsymbol{\omega}_m-\mathbf{h})+\frac{\rho_2}{2}\|\boldsymbol{\beta}_m-\mathbf{b}\|_2^2+\boldsymbol{\mu}_m^{\mathrm{T}}(\boldsymbol{\beta}_m-\mathbf{b})\right),\end{aligned} \tag{8}$$

To make the optimization amenable to decomposition on multiple (say M) learning machines, we partition the data into subsets $\{D_1,...,D_M\}$ where $D_m\subseteq\{1,...,T\}$ for $m=1,...,M$, and m indexes the machines. We can then write down the equivalent problem as (7), where ρ_1 and ρ_2 act as constant step sizes for iteration steps of our distributed algorithm. Vectors $\boldsymbol{\omega}_m$ and $\boldsymbol{\beta}_m$ are local variables of machine m which are enforced to be equal to global variables $\mathbf{h}$ and $\mathbf{b}$ respectively. The addition of quadratic terms $\|\boldsymbol{\omega}_m-\mathbf{h}\|_2^2$ and $\|\boldsymbol{\beta}_m-\mathbf{b}\|_2^2$ makes the subproblems associated with the learning at each machine strictly convex.

Now, we let $\omega:=\{\boldsymbol{\omega}_1,...\boldsymbol{\omega}_M\}$, $\beta:=\{\boldsymbol{\beta}_1,...\boldsymbol{\beta}_M\}$, $\lambda:=\{\boldsymbol{\lambda}_1,...\boldsymbol{\lambda}_M\}$ and $\mu:=\{\boldsymbol{\mu}_1,...\boldsymbol{\mu}_M\}$. An augmented Lagrangian formulation of (7) is given by (8), where λ and μ are dual variables. Notice that (8) is completely separable in $\boldsymbol{\omega}_m$, $\boldsymbol{\beta}_m$, $\mathbf{s}_m^i$ and $\boldsymbol{\xi}_m^i$. Our proposed solution approach for (8) corresponds to an iterative approach that updates, in parallel, the local (machine dependent) parameters, then shares these updates with a centralized location who updates global estimates. More specifically, each local machine updates is values of $\boldsymbol{\omega}_m,\boldsymbol{\beta}_m$, $\mathbf{s}_m$ and $\boldsymbol{\xi}_m$, then shares their local estimates of $\boldsymbol{\omega}_m$ and $\boldsymbol{\beta}_m$ with a central server. The central server uses these to update $\mathbf{h}$ and $\mathbf{b}$. The updated estimates are then broadcast by the central server to each machine, who uses them to update their own parameters in the next iteration. Overall, since (5) is a convex problem, this ADMM-based approach is guaranteed to converge to the global optimal solution in the limit, as the number of iterations becomes large. The updates to be performed for each step of our proposed d-ReLiC approach arise as solutions of subproblems of (8) each of which has a simple closed form expression (we suppress the details here due to space constraints).

4. NUMERICAL EXPERIMENT

We demonstrate the performance of d-ReLiC on a standard dataset. Since the classifiers in our approach are additionally constrained to be circularly shifted versions of each other, we expect the classification accuracy to be lower than the analogous multiclass classification approaches that impose no such constraints yet involve significantly more storage and computations. Our experimental results suggest, perhaps surprisingly, that the classification performance of our approach degrades only mildly as compared to the traditional strategies, suggesting that our approach is a viable approach in large-scale data classification problems where classifier storage space and testing complexity are critical concerns. We set $c_1=0.5$, $\eta=0.5$ and $\rho_1=\rho_2=1$ for all experiments. The selection matrix $\mathbf{S}$ is fixed to select uniformly spaced rows of the identity matrix. The data is randomly split across $M=10$ processors for d-ReLiC. For all these parameters, possibly better choices can be made through cross-validation. We used LIBSVM toolbox [21] for `MATLAB` to test the multiclass SVM schemes.

Extended Yale-B: This database [22] contains 2414 frontal-face images of 38 individuals. We use the cropped and normalized face images of size 192×168 that are captured under different illuminations for our experiments. We use 29 randomly chosen images for training and another 29 images for testing per class. We reduce the problem dimension by subsampling the vectorized images by a factor of 8 which gives 4032 dimensional vectors and solve a $K=10$ class problem. The testing phase (of all test data points) with d-ReLiC is $30\times$ ($6\times$) faster than the multiclass SVM for OvO (OvA). We also note that the classifiers associated with d-ReLiC require only 10% of the storage space that is required by the multiclass SVM classifiers. The overall comparison is summarized in Table 2.

Table 2: Performance of OvA/OvO schemes for d-ReLiC and SVM over Extended Yale-B.

	d-ReLiC (M=10)		Multiclass SVM	
	OvA	OvO	OvA	OvO
Success rate (%)	94.23	93.90	95.86	95.91
Classifier storage (kbits)	32.25	32.25	290.30	1451.52
Testing time (sec)	0.03	0.03	0.20	0.90

5. CONCLUSIONS

We introduced a novel multiclass classification method based on efficient use of a single recycled linear classifier, and described an efficient distributed procedure for its solution that allows parallel execution of updates, and is based on ADMM. Our initial simulation results suggest that the proposed approach may sacrifice little performance relative to existing multiclass classification strategies employing multiple linear hyperplane classifiers, despite providing sometimes significantly improved storage and computational complexities, making it a potentially useful approach in large-scale data settings. In addition to a more thorough performance evaluation, other extensions we plan to address in future work include employing simultaneous optimization over selection matrix $\mathbf{S}$, and extending our approach to kernel version of ReLiC.

6. REFERENCES

[1] S. Bay, "Combining nearest neighbor classifiers through multiple feature subsets," in *International Conference on Machine Learning*, 1998, pp. 37–45.

[2] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI Workshop on Empirical Methods in Artificial Intelligence*, 2001.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone, Eds., *Classification and Regression Trees*, Chapman and Hall, 1995.

[4] C. Bishop, Ed., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[5] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, pp. 273–297, 1995.

[6] R. Rifkin and A. Klautau, "Parallel networks that learn to pronounce English text," *Journal of Machine Learning Research*, pp. 101–141, 2004.

[7] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Neural Information Processing Systems*, 1998, vol. 10.

[8] S. Kumar, J. Ghosh, and M. Crawford, "Hierarchical fusion of multiple classifiers for hyperspectral data analysis," *Pattern Analysis and Applications*, vol. 5, pp. 210–220, 2002.

[9] Y. Chen, M. Crawford, and J. Ghosh, "Integrating support vector machines in a hierarchical output space decomposition framework," in *Geoscience and Remote Sensing Symposium*, 2004.

[10] T. Dietterich and G. Bakiri, "Solving multiclass learning problems via error correcting output codes," *Journal of Artificial Intelligence Research*, vol. 39, pp. 1–38, 1995.

[11] E. Allwein, R. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2001.

[12] S. Escalera, O. Pujol, and P. Radeva, "Separability of ternary codes for sparse designs of error correcting output codes," *Pattern Recognition Letters*, vol. 30, pp. 285–297, 2009.

[13] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang, "Phoneme recognition using time-delay neural networks," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 3, pp. 328–339, 1989.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] G-X. Yuan, C-H. Ho, and C-J. Lin, "Recent advances of large-scale linear classification," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2584–2603, 2012.

[16] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[18] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, June 1999.

[19] J. A. K. Suykens, J. D. Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparse approximation," *Neurocomputing*, vol. 48, no. 1-4, pp. 85–105, 2002.

[20] R. Mall and J. A. K. Suykens, "Sparse reductions for fixed-size least squares support vector machines on large scale data," in *PAKDD (1)*, 2013, pp. 161–173.

[21] C-C. Chang and C-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[22] K.C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intelligence*, vol. 27, no. 5, pp. 684–698, 2005.