Retargeted Least Squares Regression Algorithm

Xu-Yao Zhang, Lingfeng Wang, Shiming Xiang, and Cheng-Lin Liu

***Abstract*— This brief presents a framework of retargeted least squares regression (ReLSR) for multicategory classification. The core idea is to directly learn the regression targets from data other than using the traditional zero–one matrix as regression targets. The learned target matrix can guarantee a large margin constraint for the requirement of correct classification for each data point. Compared with the traditional least squares regression (LSR) and a recently proposed discriminative LSR models, ReLSR is much more accurate in measuring the classification error of the regression model. Furthermore, ReLSR is a single and compact model, hence there is no need to train two-class (binary) machines that are independent of each other. The convex optimization problem of ReLSR is solved elegantly and efficiently with an alternating procedure including regression and retargeting as substeps. The experimental evaluation over a range of databases identifies the validity of our method.**

***Index Terms*— Least squares regression (LSR), multicategory classification, retargeting.**

I. Introduction

Least squares regression (LSR) is a fundamental tool in statistics theory. Due to its effectiveness for data analysis, compact form, and efficient solution, LSR has been widely used in many machine learning problems. In addition, the derivations of many other popular models usually have strong connections to the traditional LSR model, e.g., ridge regression [16], LASSO [29], support vector machines (SVMs) [8], least squares (LS) SVM [14], [19], [28], logistic regression (LR) [17], and so on. The computational efficiency makes LSR an appealing tool for text categorization [23], biomedical analysis [3], and computer vision [18]. The LS model is also proposed as a unified framework [21] to reformulate and extend many component analysis (CA) models. Recently, various studies have extended LSR to the reproducing kernel Hilbert space to learn nonlinear classifiers [5], [6], [12], [24], [26].

Given a data set $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$ and a target set $\{\mathbf{y}_i\}_{i=1}^n \in \mathbb{R}^c$, LSR can be defined as

$$\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{n} \|\mathbf{W}^\top \mathbf{x}_i + \mathbf{b} - \mathbf{y}_i\|_2^2 + \beta \|\mathbf{W}\|_F^2 \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{d\times c}$ and $\mathbf{b} \in \mathbb{R}^c$ are to be estimated and β is a regularization parameter. The objective of LSR is to minimize the least squares (LS) loss between the regression results $\mathbf{W}^\top \mathbf{x}_i + \mathbf{b}$ and the predefined targets $\mathbf{y}_i$. In data regression analysis, each $\mathbf{y}_i$ is usually a continuous observation. However, in classification problem, only a discrete label is available for each pattern. Therefore, $\mathbf{y}_i$ is usually defined as a zero–one vector (1 for the true class and 0 for the remaining classes). However, in view of data classification, the LS loss between the regression results and the zero–one targets cannot closely reflect the classification ability of the learning machine. This drawback is more acute for multicategory classification tasks.

One strategy is to modify the loss function of LSR. For example, SVM adopts the hinge loss [8] and the squared hinge loss [4] to serve as the surrogate loss function (convex upper bound) of the misclassification error. SVM is a binary classification model. For multicategory problems, the one-against-rest, one-against-one, and ECOC [11] strategies can be used to train and combine multiple independent binary models. To treat the multicategory problem as a single and compact learning task, the negative log-exponential loss (based on soft-max) of LR [17] and the multicategory hinge loss [9] can be used to train the multicategory classifiers in a compact and joint manner.

Another strategy is to preserve the LS loss but adopt other regression targets to replace the zero–one vectors used in LSR. In [2], the regression targets are defined as the regular simplex vertices in the $\mathbb{R}^{c-1}$ (c is the number of classes) space, which are the separate points with highest degree of symmetry. The stagewise LS (SLS) model [31] uses stagewisely updated targets and solves a LS problem per stage for several stages. In this way, a bounded monotonic nonconvex loss function can be obtained, which is shown to be better than the LS loss and hinge loss. However, SLS is only proposed for binary classification. Recently, to deal with multicategory problems, Xiang *et al.* [30] proposed a technique called ε-dragging to force the regression targets of different classes moving along opposite directions, and further formulated a discriminative LSR (DLSR) model for the multicategory classification task.

In this brief, we propose a novel model called retargeted LSR (ReLSR) to directly learn the regression targets from data. ReLSR is essentially a single and compact learning machine for multicategory classification. The regression targets learned by ReLSR can guarantee each sample (data point) being correctly classified with large margin. In the learning process, ReLSR does not care about the absolute values in regression targets and only forces the relative values to satisfy a large margin constraint for the requirement of correct classification. Therefore, ReLSR is much more flexible and accurate than LSR and DLSR. The optimization problem of ReLSR is convex. An efficient alternating procedure, including regression and retargeting, is used to find the optimal solution effectively. Experiments on 15 databases demonstrated the superior performance of ReLSR compared with other benchmark multicategory models.

The idea of retargeting has also been used successfully and previously for the ranking [1] and recommendation [20] problems. The monotone retargeting (MR) as proposed in [1] searches for an order preserving (monotonic increasing) transformation of the training scores that can then be better fitted by the regressor. The Bregman divergence is used to define the loss function between the prediction function and the MR of the training scores. In this way, MR can improve the performance of ranking [1] and recommendation [20] (by combining MR with the matrix

Manuscript received October 7, 2013; revised July 5, 2014; accepted November 12, 2014. Date of publication December 2, 2014; date of current version August 17, 2015. This work was supported in part by the National Basic Research Program (973 Program) of China under Grant 2012CB316302 and in part by the National Natural Science Foundation of China under Grant 61403380 and Grant 61272331.

The authors are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: xyz@nlpr.ia.ac.cn; lfwang@nlpr.ia.ac.cn; smxiang@nlpr.ia.ac.cn; liucl@nlpr.ia.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2371492

factorization model). Different from MR, ReLSR is formulated for multicategory classification. Due to the difference between ranking and classification problems, the retargeting function of ReLSR is not required to be monotonic increasing, but only focuses on enlarging the margin between the true and most competitive false classes.

The rest of this brief is organized as follows. Section II introduces the definitions of LSR and DLSR. Section III describes the formulation and optimization procedures for ReLSR. Section IV reports the experimental results. Finally, Section V concludes this brief and discusses future work.

II. LSR and DLSR for Multiclass Classification

In this brief, we use bold fonts to represent matrices (upper-case) and vectors (lower-case), and regular fonts to represent scalars. For example: $\mathbf{X}$ is a matrix with ijth element X_{ij}, and $\mathbf{b}$ is a vector with ith element b_i. Moreover, $\mathbf{X}_{i*}$ is the ith row and $\mathbf{X}_{*j}$ is the jth column of $\mathbf{X}$.

Given n training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, 2, \ldots, c\}$ (c is the number of classes). Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^{n\times d}$ collect the n data points. The purpose is to learn a *regression matrix* $\mathbf{W} \in \mathbb{R}^{d\times c}$ and an *offset vector* $\mathbf{b} \in \mathbb{R}^c$ such that a well-defined *target matrix* $\mathbf{T}$ can be expressed approximately as

$$\mathbf{XW} + \mathbf{e}_n\mathbf{b}^\top \approx \mathbf{T} \tag{2}$$

where $\mathbf{e}_n = [1, 1, \ldots, 1]^\top \in \mathbb{R}^n$ is a vector with all 1s.

The target matrix $\mathbf{T} \in \mathbb{R}^{n\times c}$ should reflect the classification separability of each sample (row) with respect to different classes (columns). With accurate definition of $\mathbf{T}$, we can use LSR to learn the parameters of $\mathbf{W}$ and $\mathbf{b}$. In this way, each column of $\mathbf{W}$ (denoted as $\mathbf{W}_{*i}$) can be viewed as the projection axis of a linear classifier. The discriminant function of the ith class can be defined as $f_i(\mathbf{x}) = \mathbf{W}_{*i}^\top\mathbf{x} + b_i$, and a new pattern can be classified according to

$$\mathbf{x} \in \text{class } \arg\max_{i=1}^{c} f_i(\mathbf{x}). \tag{3}$$

A. LSR

An arbitrary set of c-independent vectors in $\mathbb{R}^c$ is capable of identifying c classes uniquely. Therefore, the zero–one class label vectors can be defined as the regression targets for multiclass classification. For the jth class ($j = 1, 2, \ldots, c$), define $\mathbf{f}_j = [0, \ldots, 0, 1, 0, \ldots, 0]^\top \in \mathbb{R}^c$ with only the jth element equal to one. Now, our goal is to learn a linear regression such that for the n training samples we have $\mathbf{W}^\top\mathbf{x}_i + \mathbf{b} \approx \mathbf{f}_{y_i}$.

Let $\mathbf{Y} = [\mathbf{f}_{y_1}, \mathbf{f}_{y_2}, \ldots, \mathbf{f}_{y_n}]^\top \in \mathbb{R}^{n\times c}$, the objective of LSR can be defined as

$$\text{LSR:} \quad \min_{\mathbf{W},\mathbf{b}} \|\mathbf{XW} + \mathbf{e}_n\mathbf{b}^\top - \mathbf{Y}\|_F^2 + \beta\|\mathbf{W}\|_F^2 \tag{4}$$

where $\|\cdot\|_F$ is the Frobenius norm of matrix and β is a positive regularization parameter. The first term of (4) can be viewed as a loss function, while the second term is the widely used L2 regularization to avoid overfitting.

LSR is a simple, convex, and inexpensive model, whose solutions can be efficiently obtained merely through solving linear equations (for inverse matrix problem). However, the LS loss used in (4) is nonmonotonic and boundless. Forcing the regression results to be an exact zero–one vector is not appropriate for classification tasks.

B. DLSR

To improve the classification accuracy, Xiang *et al.* [30] proposed the DLSR model for multiclass classification. The core idea is to enlarge the distance between different classes using a technique called ε-dragging to force the regression targets of different classes moving along opposite directions. Let $\mathbf{B} \in \mathbb{R}^{n\times c}$ be a constant matrix with ijth element B_{ij} defined as

$$B_{ij} = \begin{cases} +1, & \text{if } y_i = j \\ -1, & \text{otherwise} \end{cases} \tag{5}$$

the DLSR model [30] is defined as

$$\text{DLSR:} \min_{\mathbf{W},\mathbf{b},\mathbf{M}} \|\mathbf{XW} + \mathbf{e}_n\mathbf{b}^\top - \mathbf{Y} - \mathbf{B}\odot\mathbf{M}\|_F^2 + \beta\|\mathbf{W}\|_F^2 \quad \text{s.t.} \quad \mathbf{M} \geq 0 \tag{6}$$

where $\odot$ is a Hadamard product operator of matrices (i.e., element-wise product).

The target matrix of DLSR is now extended to be $\mathbf{T} = \mathbf{Y} + \mathbf{B}\odot\mathbf{M}$, where $\mathbf{M} = \{\varepsilon_{ij} \geq 0\} \in \mathbb{R}^{n\times c}$ is the ε-dragging matrix [30] that should be optimized in the learning process. Each element in $\mathbf{B}$ corresponds to a dragging direction. From the above definitions, the ijth element T_{ij} of the new target matrix $\mathbf{T}$ can be expressed as

$$T_{ij} = \begin{cases} -\varepsilon_{ij}, & \text{if } y_i \neq j \\ 1 + \varepsilon_{ij}, & \text{if } y_i = j. \end{cases} \tag{7}$$

Now, we define the regression result as $\mathbf{R} = \mathbf{XW} + \mathbf{e}_n\mathbf{b}^\top \in \mathbb{R}^{n\times c}$ and let R_{ij} be the ijth element of $\mathbf{R}$. By minimizing $(R_{ij} - T_{ij})^2$, the optimized target matrix of DLSR can be expressed as

$$T_{ij} = \begin{cases} \min(R_{ij}, 0), & \text{if } y_i \neq j \text{ (false class)} \\ \max(R_{ij}, 1), & \text{if } y_i = j \text{ (true class).} \end{cases} \tag{8}$$

Therefore, the loss function (first term) of (6) can be rewritten as $\sum_{i=1}^n\sum_{j=1}^c \mathcal{L}_{ij}$, where

$$\mathcal{L}_{ij} = (R_{ij} - T_{ij})^2 = \begin{cases} [\max(R_{ij}, 0)]^2, & \text{if } y_i \neq j \text{ (false class)} \\ [\min(R_{ij} - 1, 0)]^2, & \text{if } y_i = j \text{ (true class).} \end{cases} \tag{9}$$

From the definition of $\mathcal{L}_{ij}$, we can find that (Fig. 1): the regression result for the false class should be smaller than zero, and the regression result for the true class should be larger than one, otherwise it will cause a nonzero loss. In this way, each sample can be classified correctly with a large margin between the true and false classes. Actually, by defining target matrix as $\mathbf{T} = \mathbf{Y} + \mathbf{B}\odot\mathbf{M}$, the new loss function (9) is the squared hinge loss [4] that is more suitable for classification tasks than the LS loss used in LSR.

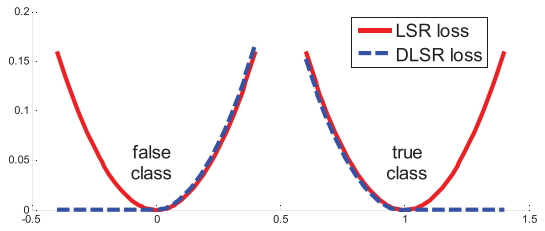

Fig. 1. Loss function comparison of LSR and DLSR.

III. ReLSR for Multiclass Classification

In this section, we first introduce the formulation of the new model, then we compare ReLSR with other models. At last, we describe the optimization procedures for solving ReLSR.

TABLE I
LOSS FUNCTIONS AND REGRESSION TARGETS FOR LSR, DLSR, AND RELSR ON DIFFERENT SAMPLES

	class	$\mathbf{y}$	regression result	margin≥ 1	LSR	DLSR	ReLSR
$\mathbf{x}_1$	1	$[1,0,0]$	$[1.5,0,0]$	Yes	loss=0.25	loss=0.00 target=$[1.5,0,0]$	loss=0.00 target=$[1.5,0,0]$
$\mathbf{x}_2$	1	$[1,0,0]$	$[1,-0.5,-0.5]$	Yes	loss=0.50	loss=0.00 target=$[1,-0.5,-0.5]$	loss=0.00 target=$[1,-0.5,-0.5]$
$\mathbf{x}_3$	2	$[0,1,0]$	$[0.5,1.5,0.5]$	Yes	loss=0.75	loss=0.50 target=$[0,1.5,0]$	loss=0.00 target=$[0.5,1.5,0.5]$
$\mathbf{x}_4$	2	$[0,1,0]$	$[-0.5,1.5,0.5]$	Yes	loss=0.75	loss=0.25 target=$[-0.5,1.5,0]$	loss=0.00 target=$[-0.5,1.5,0.5]$
$\mathbf{x}_5$	3	$[0,0,1]$	$[0.2,0.2,0.8]$	No	loss=0.12	loss=0.12 target=$[0,0,1]$	loss=0.11 target=$[0.0667,0.0667,1.0667]$
$\mathbf{x}_6$	3	$[0,0,1]$	$[-0.2,0.2,0.6]$	No	loss=0.24	loss=0.20 target=$[-0.2,0,1]$	loss=0.18 target=$[-0.2,-0.1,0.9]$

A. Problem Formulation

Aiming at improving the performance for multiclass classification, we propose to learn the regression target matrix directly from data. To guarantee each sample being correctly classified with large margin, the target matrix $\mathbf{T} \in \mathbb{R}^{n\times c}$ should be optimized in the learning process with a constraint that for each sample (row), the margin between the targets of true and false classes should be larger than one, i.e., $T_{i,y_i} - \max_{j\neq y_i} T_{i,j} \geq 1$. Therefore, the learning problem can be formulated as an optimization problem

$$\text{ReLSR:} \quad \min_{\mathbf{W},\mathbf{b},\mathbf{T}} \|\mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{b}^\top - \mathbf{T}\|_F^2 + \beta\|\mathbf{W}\|_F^2 \quad \text{s.t.} \quad T_{i,y_i} - \max_{j\neq y_i} T_{i,j} \geq 1, \quad i = 1,2,\ldots,n. \tag{10}$$

We denote model (10) as ReLSR, because in the learning process, the target matrix is repeatedly updated to minimize the regression error with regression and retargeting as substeps (see the following sections). The target matrix of ReLSR is directly learned from data with a constraint closely reflecting the classification separability of each data point. Therefore, ReLSR should be much more flexible and accurate than DLSR and LSR.

B. Comparing ReLSR, DLSR, and LSR

Table I shows the performance of different models for six data points. The regression results are shown in the fourth column (assuming $\mathbf{W}$ and $\mathbf{b}$ being fixed). The fifth column shows whether or not the regression result satisfies margin $\geq$1 (margin is defined as the minimal difference between the regression results for the true and false classes). The last three columns are the loss functions and regression targets for each data point. The loss function is defined as the difference (Euclidean distance) between regression results and targets.

Consider $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$ in Table I, all of them can be classified correctly with margin ≥ 1, therefore the loss functions of them should be zero. The regression results of $\mathbf{x}_5$ and $\mathbf{x}_6$ cannot fulfill the constraint of margin ≥ 1, therefore the loss functions of them should be larger than zero. From Table I, we can find that both LSR and DLSR cannot satisfy the above requirements, only ReLSR can give appropriate loss functions for different data points. As shown in Fig. 1, LSR forces the regression results to be around 0 for false classes and around 1 for true classes, while DLSR forces the regression results to be smaller than 0 for false classes and larger than 1 for true classes. Both LSR and DLSR use two *absolute values* (0 and 1) to define their regression targets. Contrarily, the target matrix of ReLSR is learned by only focusing on the *relative values* corresponding to different classes for guaranteeing correct classification with large margin. Therefore, the target matrix of ReLSR is much more accurate than LSR and DLSR in measuring the classification performance (loss function) of the learning machine.

Another important advantage is that ReLSR is a compact and single machine for multiclass classification. From the definitions of LSR (4) and DLSR (6), we can find that the learning problem can be decomposed into c equivalent independent subproblems. The kth subproblem corresponds to the learning of $\mathbf{W}_{*k}$, b_k, and $\mathbf{M}_{*k}$ (the classifier parameters for the kth class). That means LSR and DLSR are equivalent to the learning of c one-against-rest binary regression models. On the contrary, because the columns of $\mathbf{T}$ in ReLSR (10) are now dependent with each other (due to the large margin constraint), the objective of ReLSR cannot be decomposed into independent subproblems. With our formulation, the c subproblems are grouped together to share a unique learning model. Therefore, ReLSR is essentially a compact and single machine for multiclass classification, which should be more accurate and flexible than the one-against-rest models.

C. Comparing ReLSR and Multiclass Hinge Loss

ReLSR is related to but different from the multiclass SVM (MC-SVM) model [9]. Let $\mathbf{R} = \mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{b}^\top \in \mathbb{R}^{n\times c}$ denotes the classifier outputs. The multiclass hinge loss [9] is defined as

$$\xi_i = \max\{0, 1 - R_{i,y_i} + \max_{j\neq y_i} R_{ij}\} \qquad \forall i = 1,\ldots,n \tag{11}$$

which reflects the margin between the true and most competitive false classes. The classifier is then trained to minimize the empirical loss (L1 or L2) as

$$\text{L1:} \ \min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{n} \xi_i + \beta\|\mathbf{W}\|_F^2, \quad \text{L2:} \ \min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{n} \xi_i^2 + \beta\|\mathbf{W}\|_F^2. \tag{12}$$

These formulations are based on modification of loss functions other than redefinition of target matrix in the LSR framework. Both of them are not equivalent to ReLSR. The model of (10) can be viewed as a reformulation of the multiclass hinge loss into the LSR framework.

D. Solving the Optimization Model

The objective of the ReLSR model (10) is to minimize a convex quadratic function and subject to a linear constraint, hence the model of (10) is jointly convex for all the parameters. Therefore, the iterative alternating optimization procedures can be used to find the unique optimal solution efficiently.

1) Regression Problem (Fix $\mathbf{T}$, and Optimize $\mathbf{W}$ and $\mathbf{b}$): By fixing the target matrix $\mathbf{T}$, the problem of (10) is changed to the classical LSR problem

$$\text{Regression:} \quad \min_{\mathbf{W},\mathbf{b}} \|\mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{b}^\top - \mathbf{T}\|_F^2 + \beta\|\mathbf{W}\|_F^2 \tag{13}$$

which can be solved with a closed-form solution [30]. Let $\mathbf{H} = \mathbf{I}_n - \mathbf{e}_n\mathbf{e}_n^\top/n$ and $\mathbf{I}_n$ is a $n \times n$ identity matrix. The optimal solution can be calculated as

$$\mathbf{W} = (\mathbf{X}^\top\mathbf{H}\mathbf{X} + \beta\mathbf{I}_d)^{-1}\mathbf{X}^\top\mathbf{H}\mathbf{T}, \ \ \mathbf{b} = \frac{(\mathbf{T} - \mathbf{X}\mathbf{W})^\top\mathbf{e}_n}{n} \tag{14}$$

where $\mathbf{I}_d$ is a $d \times d$ identity matrix. To prove the above results, we denote $g(\mathbf{W},\mathbf{b}) = \|\mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{b}^\top - \mathbf{T}\|_F^2 + \beta\|\mathbf{W}\|_F^2$, and then

we have

$$\frac{\partial g(\mathbf{W},\mathbf{b})}{\partial \mathbf{b}} = 0 \Rightarrow \mathbf{W}^\top\mathbf{X}^\top\mathbf{e}_n + \mathbf{b}\mathbf{e}_n^\top\mathbf{e}_n - \mathbf{T}^\top\mathbf{e}_n = 0$$
$$\Rightarrow \mathbf{b} = \frac{(\mathbf{T}-\mathbf{X}\mathbf{W})^\top\mathbf{e}_n}{n}.$$

Furthermore, we have

$$\begin{aligned}\frac{\partial g(\mathbf{W},\mathbf{b})}{\partial \mathbf{W}} &= 0\\ &\Rightarrow \mathbf{X}^\top(\mathbf{X}\mathbf{W}+\mathbf{e}_n\mathbf{b}^\top-\mathbf{T})+\beta\mathbf{W}=0\\ &\Rightarrow \mathbf{X}^\top\left(\mathbf{X}\mathbf{W}+\frac{1}{n}\mathbf{e}_n\mathbf{e}_n^\top(\mathbf{T}-\mathbf{X}\mathbf{W})-\mathbf{T}\right)+\beta\mathbf{W}=0\\ &\Rightarrow \mathbf{X}^\top\mathbf{H}\mathbf{X}\mathbf{W}-\mathbf{X}^\top\mathbf{H}\mathbf{T}+\beta\mathbf{W}=0\\ &\Rightarrow \mathbf{W}=(\mathbf{X}^\top\mathbf{H}\mathbf{X}+\beta\mathbf{I}_d)^{-1}\mathbf{X}^\top\mathbf{H}\mathbf{T}.\end{aligned}$$

Therefore, (14) is the optimal solution for (13).

The hyperparameter β in the regression problem (13) is a tradeoff parameter to avoid overfitting. Considering the data scaling problem and motivated from the solution of (14), we suggest to set β as

$$\beta = \widehat{\beta}\frac{1}{d}\mathrm{tr}(\mathbf{X}^\top\mathbf{H}\mathbf{X}) \tag{15}$$

where $\mathrm{tr}(\cdot)$ is the trace of a matrix. Now, we can use cross validation to efficiently and effectively select $\widehat{\beta}$.[1]

2) Retargeting Problem (Fix $\mathbf{W}$ *and* $\mathbf{b}$*, and Optimize* $\mathbf{T}$*):* By fixing the regression matrix $\mathbf{W}$ and offset vector $\mathbf{b}$, the problem of (10) is changed to a retargeting problem

$$\text{Retargeting:}\quad \min_{\mathbf{T}} \|\mathbf{X}\mathbf{W}+\mathbf{e}_n\mathbf{b}^\top-\mathbf{T}\|_F^2 = \|\mathbf{R}-\mathbf{T}\|_F^2$$
$$\text{s.t.}\quad T_{i,y_i} - \max_{j\neq y_i} T_{i,j} \geq 1,\quad i=1,2,\ldots,n. \tag{16}$$

Here, we use $\mathbf{R}\in\mathbb{R}^{n\times c}$ to denote the regression result $\mathbf{X}\mathbf{W}+\mathbf{e}_n\mathbf{b}^\top$, and use T_{ij} to denote the ijth element of $\mathbf{T}$. In this problem, the target matrix $\mathbf{T}\in\mathbb{R}^{n\times c}$ is optimized to minimize the regression error, and each element of $\mathbf{T}$ should satisfy the large margin constraint for the requirement of correct classification.

The retargeting problem (16) is a convex constrained quadratic programming (QP) problem, which can be decomposed into n independent subproblems. Each subproblem corresponds to the learning of one row of $\mathbf{T}$. Let $\mathbf{r}=[r_1,r_2,\ldots,r_c]\in\mathbb{R}^{1\times c}$ be one row of $\mathbf{R}$ (regression result), and $\mathbf{t}=[t_1,t_2,\ldots,t_c]\in\mathbb{R}^{1\times c}$ be the same row of $\mathbf{T}$ (target matrix). The true class index for this row (data point) is denoted by k (e.g., y_i for row i). The problem (16) can be decomposed into n subproblems with a general form as

$$\min_{\mathbf{t}} \|\mathbf{r}-\mathbf{t}\|_2^2 = \sum_{i=1}^{c}(r_i-t_i)^2 \quad \text{s.t.}\quad t_k - \max_{i\neq k} t_i \geq 1. \tag{17}$$

Formally, we only need to solve (17), and after that, (16) can be solved row-by-row with the same procedure.

To solve problem (17), we introduce another variable $\mathbf{v}=[v_1,v_2,\ldots,v_c]\in\mathbb{R}^{1\times c}$, where

$$v_i = r_i + 1 - r_k \qquad \forall i = 1,2,\ldots,c. \tag{18}$$

Here, $v_i\leq 0$ means class i and class k (true class) satisfy the margin ≥ 1 and $v_i>0$ means they violate margin constraint. Suppose the target for the true class is a small modification of the regression result $t_k=r_k+\triangle$, where the step parameter $\triangle$ should be optimized in the learning process. Because the constraint of (17) is equivalent to $t_k-t_i\geq 1,\forall i\neq k$, we can optimize the targets for false classes ($\forall i\neq k$) one-by-one as (by fixing $t_k=r_k+\triangle$)

$$\min_{t_i}(r_i-t_i)^2 \quad \text{s.t.}\quad r_k+\triangle-t_i\geq 1. \tag{19}$$

[1]In our experiments, we select $\widehat{\beta}$ from the interval of $[0,1]$.

Algorithm 1 ReLSR Algorithm

Input:
data $\{(\mathbf{x}_i,y_i)\}_{i=1}^n$ where $\mathbf{x}_i\in\mathbb{R}^d$, $y_i\in\{1,2,\ldots,c\}$
hyper-parameter: β, iterNum
1: data matrix $\mathbf{X}=[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n]^\top\in\mathbb{R}^{n\times d}$
2: $\mathbf{H}=\mathbf{I}_n-\frac{1}{n}\mathbf{e}_n\mathbf{e}_n^\top$ and $\mathbf{U}=(\mathbf{X}^\top\mathbf{H}\mathbf{X}+\beta\mathbf{I}_d)^{-1}\mathbf{X}^\top\mathbf{H}$
3: $\mathbf{T}=\{T_{ij}\}\in\mathbb{R}^{n\times c}$ where $T_{ij}=\begin{cases}1, & \text{if } y_i=j\\ 0, & \text{otherwise.}\end{cases}$
4: **for** iter $=1\rightarrow$ iterNum **do**
5: **regression:** $\mathbf{W}=\mathbf{U}\mathbf{T}$ and $\mathbf{b}=\frac{(\mathbf{T}-\mathbf{X}\mathbf{W})^\top\mathbf{e}_n}{n}$
6: **retargeting:** $\mathbf{R}=\mathbf{X}\mathbf{W}+\mathbf{e}_n\mathbf{b}^\top\in\mathbb{R}^{n\times c}$
7: **for** $i=1\rightarrow n$ **do**
8: $\mathbf{T}_{i*}=\text{Retargeting}(\mathbf{R}_{i*},y_i)$ according to Algorithm 2
9: **end for**
10: **end for**
Output: $\mathbf{W}\in\mathbb{R}^{d\times c}$ and $\mathbf{b}\in\mathbb{R}^c$

By solving this *single-variable constrained QP* problem, we can get $t_i=r_i+\min(\triangle-v_i,0),\forall i\neq k$.

Now, we can redefine target vector $\mathbf{t}$ in (17) with only one parameter $\triangle$ (need to be optimized)

$$t_i = \begin{cases} r_i+\triangle, & \text{if } i=k\\ r_i+\min(\triangle-v_i,0), & \text{if } i\neq k.\end{cases} \tag{20}$$

According to (20), problem (17) can be changed to

$$\min_{\triangle}\ g(\triangle)=\triangle^2+\sum_{i\neq k}(\min(\triangle-v_i,0))^2. \tag{21}$$

Furthermore, let $g'(\triangle)=\partial g(\triangle)/\partial\triangle$, we have

$$g'(\triangle)=\triangle+\sum_{i\neq k}\min(\triangle-v_i,0). \tag{22}$$

Now, the last problem is to solve the equation of $g'(\triangle)=0$. Because $g'(\triangle)$ is a *monotone increasing piecewise linear* function, the optimal $\triangle$ can be calculated as[2]

$$\triangle = \frac{\sum_{i\neq k} v_i\mathbb{I}(g'(v_i)>0)}{1+\sum_{i\neq k}\mathbb{I}(g'(v_i)>0)} \tag{23}$$

where $\mathbb{I}(\cdot)=1$ when the condition in brackets is true and otherwise $\mathbb{I}(\cdot)=0$. After computing $\triangle$, the optimal target $\mathbf{t}$ can be obtained via (20).

E. Algorithm of ReLSR

On the basis of the above analyses, we develop an iterative method to solve the primal problem of (10). Algorithm 1 lists the steps for the ReLSR problem, and Algorithm 2 lists the steps for the retargeting problem.

In step 3 of Algorithm 1, the initial target matrix is defined as the zero–one matrix. After that, the regression (step 5) and retargeting (steps 6–9) subproblems are solved repeatedly to get the optimal solution for ReLSR (10). Because the ReLSR problem is jointly and strictly convex, the alternating procedures are guaranteed to find the unique optimal solution.

To save the computational cost in the regression step (14), a matrix $\mathbf{U}$ is calculated outside the iterations in step 2, and in each iteration $\mathbf{W}$ can be obtained via step 5.

[2]The proof is in the Appendix.

Algorithm 2 Retargeting Algorithm

Input:

regression result: $\mathbf{r} = [r_1, r_2, \ldots, r_c] \in \mathbb{R}^{1\times c}$

true class index: k

1: set $\upsilon_i = r_i + 1 - r_k, \forall i = 1, 2, \ldots, c$

2: set $\triangle = 0$ and $m = 0$

3: **for** $i = 1 \rightarrow c$ $(i \neq k)$ **do**

4: **if** $g'(\upsilon_i) = \upsilon_i + \sum_{j\neq k} \min\left(\upsilon_i - \upsilon_j, 0\right) > 0$ **then**

5: set $\triangle = \triangle + \upsilon_i$ and $m = m + 1$

6: **end if**

7: **end for**

8: set $\triangle = \frac{\triangle}{1+m}$

9: $t_i = \begin{cases} r_i + \triangle, & \text{if } i = k \\ r_i + \min(\triangle - \upsilon_i, 0), & \text{if } i \neq k \end{cases}$

Output: target vector $\mathbf{t} = [t_1, t_2, \ldots, t_c] \in \mathbb{R}^{1\times c}$

In the retargeting problem (steps 6–9 in Algorithm 1), the regression result matrix $\mathbf{R}$ is first calculated. After that, each row of the target matrix $\mathbf{T}_{i*}$ is obtained via the retargeting function in Algorithm 2 with $\mathbf{R}_{i*}$ and y_i as inputs.

In Algorithm 2 of retargeting, variable υ_i (18) is calculated in step 1, and the optimal $\triangle$ (23) is calculated in steps 2–8. Finally, the target vector (20) is obtained in step 9.

The computational complexity in step 2 of Algorithm 1 is $O(2nd^2 + d)$. Furthermore, the main computational costs of Algorithm 1 come from steps 5–9. The complexity of step 5 is $O(ndc)$, while the complexity of step 6 is also $O(ndc)$. The complexity of steps 7–9 is $O(nc)$. Suppose the number of iterations to be m, the total computational complexity of ReLSR is about $O(2nd^2 + d + mnc(2d + 1))$. Furthermore, the retargeting procedure (steps 7–9 in Algorithm 1) is independent for each row (sample), and therefore, can be easily and efficiently parallelized.

F. Convergence Analysis

To analyze the convergence of Algorithm 1, we denote the objective function in (10) by $G(\mathbf{W}, \mathbf{b}, \mathbf{T})$. Denote the value of the objective function at the $(t-1)$th iteration as $G(\mathbf{W}^{t-1}, \mathbf{b}^{t-1}, \mathbf{T}^{t-1})$. During the tth iteration, we first fix the target matrix $\mathbf{T}^{t-1}$ and solve the subproblem of $\min_{\mathbf{W},\mathbf{b}} G(\mathbf{W}, \mathbf{b}, \mathbf{T}^{t-1})$. Solving it via (13) yields the optimal solution $(\mathbf{W}^t, \mathbf{b}^t)$ at the tth iteration. Since this subproblem is convex, naturally we have

$$G(\mathbf{W}^{t-1}, \mathbf{b}^{t-1}, \mathbf{T}^{t-1}) \geq G(\mathbf{W}^t, \mathbf{b}^t, \mathbf{T}^{t-1}). \tag{24}$$

Next, by fixing $(\mathbf{W}^t, \mathbf{b}^t)$, we solve the subproblem of $\min_{\mathbf{T}} G(\mathbf{W}^t, \mathbf{b}^t, \mathbf{T})$ with the constraint in (10). Solving it via (16) yields the optimal $\mathbf{T}^t$. Due to the convexity of this subproblem, it follows that

$$G(\mathbf{W}^t, \mathbf{b}^t, \mathbf{T}^{t-1}) \geq G(\mathbf{W}^t, \mathbf{b}^t, \mathbf{T}^t). \tag{25}$$

Combining (24) and (25) together, we get

$$G(\mathbf{W}^{t-1}, \mathbf{b}^{t-1}, \mathbf{T}^{t-1}) \geq G(\mathbf{W}^t, \mathbf{b}^t, \mathbf{T}^t). \tag{26}$$

In this way, we can conclude that Algorithm 1 can monotonically decrease the value of $G(\mathbf{W}, \mathbf{b}, \mathbf{T})$. Since (10) is a joint convex model, the alternating optimization procedures used in Algorithm 1 will find the unique optimal solution of ReLSR.

IV. Experiments

In this section, we compare ReLSR with six multicategory models on a range of different databases. We first describe the data sets used for evaluation and then introduce the parameter settings used in our experiments, at last we report the experimental results and analysis.

TABLE II
Brief Descriptions of the Data Sets for Classification

	Classes	Features	Total Num.	Train Num.
iris	3	4	150	60
svmguide2	3	20	391	156
DNA	3	180	3186	1274
vehicle	4	18	846	338
glass	6	9	214	86
vowel	11	10	990	396
AT&T	40	644	400	160
Umist	20	2576	575	230
Yale	15	1024	165	66
Coil20	20	256	1440	576
USPS	10	256	2000	800
Cora-OS	4	6737(300)	1246	499
WebKB-CL	7	4134(300)	827	331
WebKB-WT	7	4165(300)	1166	466
WebKB-WC	7	4189(300)	1210	484

A. Data Sets

We used 15 databases to evaluate the performance of the proposed method. Table II describes the information of these data sets. The first six data sets (*iris*, *svmguide2*, *DNA*, *vehicle*, *glass*, and *vowel*) are taken from the LIBSVM machine learning data repository.[3] The next five databases (*AT&T*, *Umist*, *Yale*, *Coil20*, and *USPS*) are for image classification (face, object, and digit). The last four databases (*Cora-OS*, *WebKB-CL*, *WebKB-WT*, and *WebKB-WC*) are for information extraction and retrieval.

AT&T is a face database of 40 persons, and each person has 10 gray images with different expressions and facial details.[4] The size of each image is 92×112 that is further resized to be 28×23. Thus, the source dimensionality is 644. *Umist* contains the face images of 20 different persons.[5] Each image is resized to be 56×46. The source dimensionality is 2576. *Yale* face database contains 165 gray scale images of 15 individuals.[6] Each image is resized to be 32×32. The source dimensionality is 1024. *Coil20* includes 20 objects, each of which has 72 gray images, which are taken from different view directions.[7] Each image is resized to be 16×16. Thus, the dimensionality is 256. *USPS* is a database with 10 digits.[8] For each digit, 200 images with 16×16 pixels are randomly selected to construct a data set. The dimensionality is 256.

Cora-OS is a subset containing the research papers about operating system [25]. *WebKB* databases contain a subset consisting of about 3200 web pages from computer science departments of three schools (Cornell, Washington, and Wisconsin).[9] For the last four data sets, principal component analysis is used to project them into 300D subspace.[10]

B. Parameter Settings

The proposed ReLSR model (10) is compared with the traditional LSR (4), the recently proposed DLSR (6), L1-SVM with hinge

[3] http://www.csie.ntu.edu.tw/%7Ecjlin/libsvmtools/datasets/

[4] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[5] http://images.ee.umist.ac.uk/danny/database.html

[6] http://cvc.yale.edu/projects/yalefaces/yalefaces.html

[7] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[8] http://www.kernel-machines.org/data

[9] http://www.cs.cmu.edu/%7Ewebkb/

[10] Better accuracies can be achieved using 300D subspace compared with the 200D subspace used in [30].

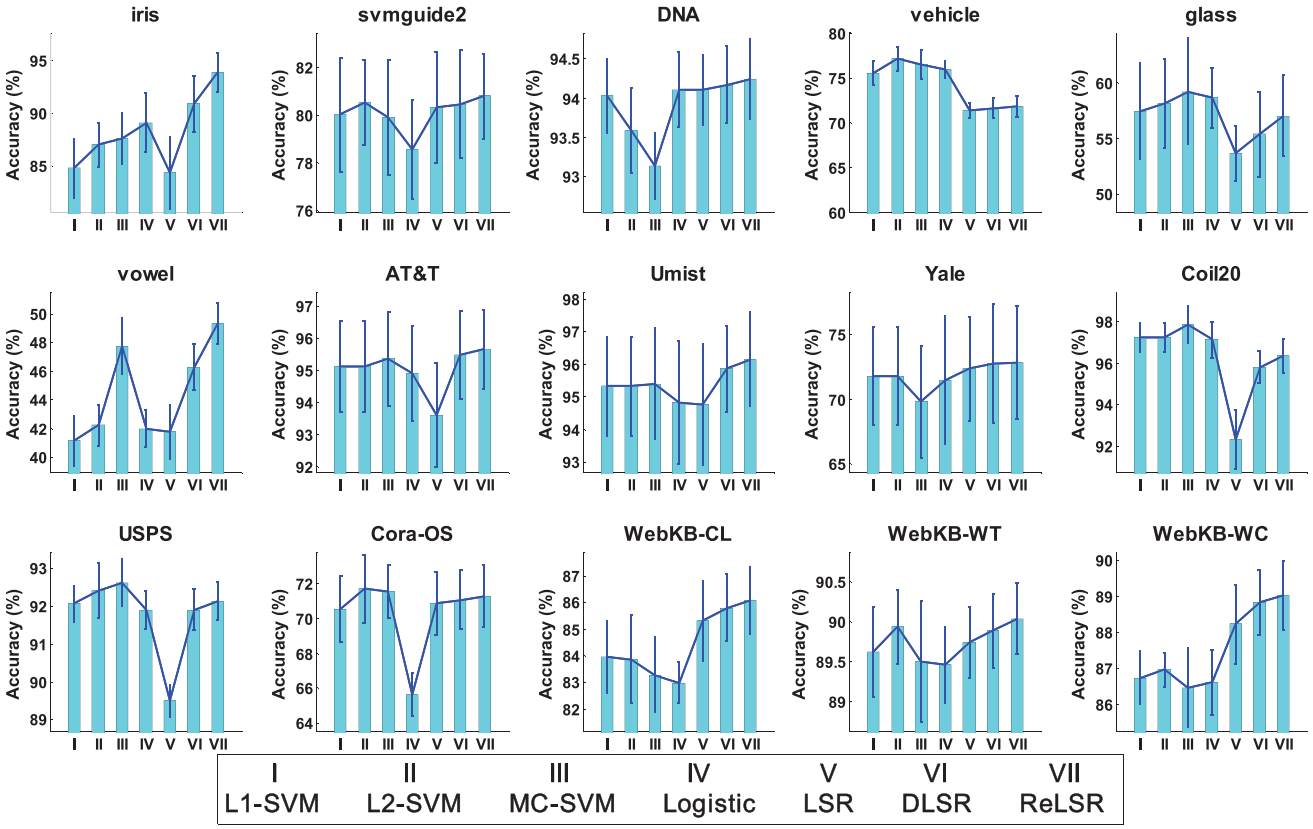

Fig. 2. Classification accuracy and standard deviation comparison of different models.

loss [8], L2-SVM with squared hinge loss [4], logistic regression (LR) [17], and the multiclass SVM (MC-SVM) with multiclass hinge loss [9]. The one-against-rest rule is employed in both L1-SVM and L2-SVM to fulfill the training tasks for multiclass classification. The implementations of L1-SVM, L2-SVM, LR, and MC-SVM are all included in the LIBLINEAR software [13].

In our experiments, the maximum number of iterations in Algorithm 1 is set as 30. ReLSR has only one parameter β to be selected. Different from the work in [30] where a candidate set of parameters for β is fixed for all of the data sets, here we define β as (15) and use tenfold cross validation to select $\widehat{\beta}$ from $[0, 1]$ (uniform sampling with 0.05 as steps). This strategy is also used to select the regularization parameter for LSR and DLSR. For L1-SVM, L2-SVM, LR, and MC-SVM, there exists an important regularization parameter C in LIBLINEAR. We use cross-validation approach to select it from the candidate set of $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. Finally, by fixing the selected parameters ($\widehat{\beta}$ or C), all the models are evaluated on the databases in Table II with $10\times$ randomly partition of the training and testing sets (40% for training and 60% for testing).

C. Experimental Results

Fig. 2 shows the average accuracy and the standard deviation of different models. We can find that in most cases, LSR is inferior to SVM-based methods (L1-SVM, L2-SVM, and MC-SVM). This is because the LS loss between the regression results and the zero–one targets is not a suitable criterion for classifier training. The hinge loss used in SVM-based methods is more accurate in measuring the classification error. However, with the redefinition of the target matrix, DLSR and ReLSR can significantly improve the classification accuracy of LSR, which makes them comparable or even better than the SVM-based methods (e.g., Fig. 2: *iris*, *vowel*, *Umist*, *WebKB-CL*, *WebKB-WT*, *WebKB-WC*, and so on). This justifies the correctness of modifying the regression targets in improving the classification accuracy of the regression model.

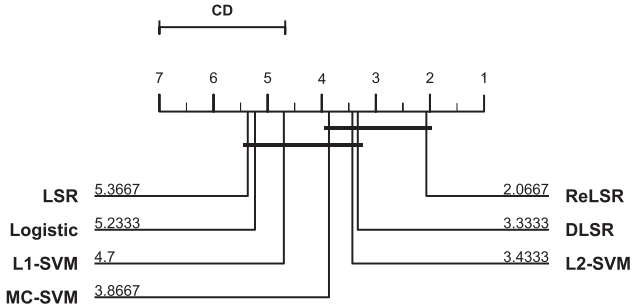

Fig. 3. CD diagram of different models.

It is also verified that ReLSR is better than LR. LR adopts a soft-max based probabilistic criterion to train the classifier, and a highly nonlinear optimization problem should be solved. Contrarily, in ReLSR, only two alternating QP problems are solved, which can be efficiently optimized with a closed-form solution in the regression step and a low-computational complexity solution in the retargeting step.

Furthermore, from Fig. 2, we can find that ReLSR outperforms LSR and DLSR significantly. For all the 15 data sets, ReLSR can consistently improve the classification accuracy over LSR and DLSR. This justifies that the target matrix learned by ReLSR is more flexible and accurate in measuring the classification error than the traditional zero–one target matrix of LSR and the newly proposed target matrix of DLSR. Different from LSR and DLSR, ReLSR learns the target matrix directly from data and only cares about the relative values in

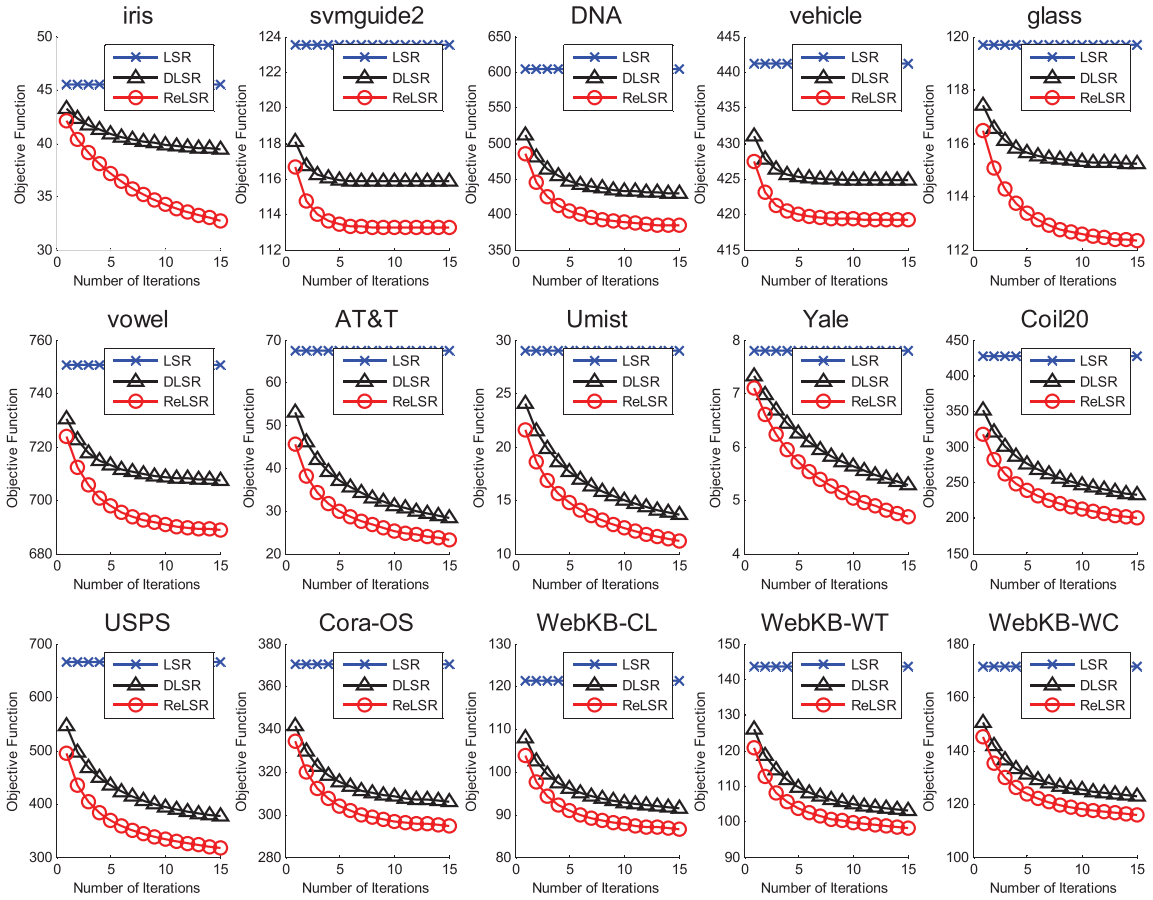

Fig. 4. Convergence behavior: the objective functions with respect to the number of iterations for LSR, DLSR, and ReLSR.

target matrix for the requirement of correct classification for each data point with large margin. Therefore, better generalization performance can be achieved with the new formulation of ReLSR.

D. Statistical Significance

The Friedman test [10] is used to compare the average ranks of different algorithms. The null hypothesis states that all the algorithms are equivalent, and so, their ranks should be equal. If the null hypothesis is rejected, we can proceed with a post-hoc test (the Nemenyi test) to find out which algorithms significantly differ. Specifically, the performance of two algorithms is significantly different if their average ranks differ by at least the critical difference (CD) [10].

In Fig. 2, we have a total of 7 models with 15 evaluations. Fig. 3 shows the CD diagram for the seven models, where the average rank of each compared model is marked along the axis. The axis is turned so that the lowest (best) ranks are to the right. Groups of models that are not significantly different are connected with a thick line. We can find that (Fig. 3): LSR has the highest rank, however, with the redefinition of the regression target matrix, DLSR and ReLSR can significantly improve the performance, and ReLSR achieves the lowest rank (best performance) among all the models.

E. Convergence Analysis

To analyze the convergence behavior of the proposed algorithm, we show in Fig. 4 the objective functions of LSR (4), DLSR (6), and ReLSR (10) with respect to the number of iterations, and each iteration corresponds to a regression and a retargeting step. Because the target matrix of LSR is fixed as a zero–one matrix, the objective function of LSR is not changed during iterations. The optimization of DLSR is very similar to ReLSR, except that DLSR uses a different retargeting strategy [30]. We can find that for all the 15 databases, the objective functions of ReLSR are dropped very fast within only a few number of iterations. This justifies the effectiveness of the alternating optimization algorithm in solving the ReLSR problem.

F. Discussion of LSR, DLSR, and ReLSR

Furthermore, from Fig. 4, we can also find that the objective functions of ReLSR are much smaller than those of LSR and DLSR. This is because the searching spaces of target matrices for different models have the relationship of LSR $\subseteq$ DLSR $\subseteq$ ReLSR. That means the optimal target matrices of LSR and DLSR can satisfy the large margin constraint used in ReLSR, but not vice versa. In other words, the optimal solutions of LSR and DLSR are covered in the searching space of ReLSR. Therefore, the target matrix of ReLSR should be much more flexible, and smaller regression error (objective function) can be achieved by ReLSR.

V. Conclusion

In this brief, a simple and effective framework of ReLSR is proposed for multicategory classification. The core idea is to directly learn the regression target matrix from data by focusing on the relative values for the requirement of correct classification with large margin. Two efficient subproblems regression and retargeting are used to find the unique optimal solution of ReLSR. Experimental results identify the superior performance of ReLSR against other benchmark methods.

The framework of ReLSR can be easily combined with other machine learning tricks, such as L_{21} norm based feature

selection [30], dropout training [7], [15], and so on. Using retargeting for nonlinear classifier learning, such as kernelization of ReLSR based on explicit feature mapping [22], [27] or other method [5] is also an important future direction. The idea of direct learning the regression targets from data (retargeting) can be hopefully extended to many other problems, such as manifold learning and semisupervised learning. Other types of classifiers (such as artificial neural networks) can also be improved by retargeting when they adopt the mean squared error as the optimization criterion.

APPENDIX

Here, we prove that (23) is the solution of $g'(\triangle) = 0$. Let x being the optimal solution that means $g'(x) = 0$. It is easy to prove that $g'(\cdot)$ in (22) is a monotone increasing function. Therefore, $g'(v_i) > 0 \Leftrightarrow v_i > x$. Now, we have

$$\begin{aligned} g'(x) &= x + \sum_{i \neq k} \min(x - v_i, 0) \\ &= x + \sum_{i \neq k} (x - v_i) \mathbb{I}(v_i > x) \\ &= x + \sum_{i \neq k} (x - v_i) \mathbb{I}(g'(v_i) > 0). \end{aligned} \tag{27}$$

Hence, by solving $g'(x) = 0$, we have $x = \sum_{i \neq k} v_i \mathbb{I}(g'(v_i) > 0) / [1 + \sum_{i \neq k} \mathbb{I}(g'(v_i) > 0)]$.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful suggestions to improve the quality of this brief.

REFERENCES

[1] S. Acharyya, O. Koyejo, and J. Ghosh, "Learning to rank with Bregman divergences and monotone retargeting," in *Proc. Uncertainty Artif. Intell. (UAI)*, 2012.

[2] S. An, W. Liu, and S. Venkatesh, "Face recognition using kernel ridge regression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2007, pp. 1–8.

[3] W. L. Anderson, "Optical processing for least mean-squares analysis of biomedical patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, no. 5, pp. 458–463, Sep. 1980.

[4] K.-W. Chang, C.-J. Hsieh, and C.-J. Lin, "Coordinate descent method for large-scale L2-loss linear support vector machines," *J. Mach. Learn. Res.*, vol. 9, pp. 1369–1398, Jan. 2008.

[5] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 22–32, Jan. 2012.

[6] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, "Quantized kernel recursive least squares algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 9, pp. 1484–1491, Sep. 2013.

[7] N. Chen, J. Zhu, J. Chen, and B. Zhang, "Dropout training for support vector machines," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2014.

[8] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

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

[10] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[11] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *J. Artif. Intell. Res.*, vol. 2, no. 1, pp. 263–286, 1995.

[12] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, 2008.

[14] T. van Gestel *et al.*, "Benchmarking least squares support vector machine classifiers," *Mach. Learn.*, vol. 54, no. 1, pp. 5–32, 2004.

[15] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. (2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: http://arxiv.org/abs/1207.0580

[16] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[17] D. W. Hosmer, Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. New York, NY, USA: Wiley, 2013.

[18] D. Huang, R. S. Cabral, and F. De la Torre, "Robust regression," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2012, pp. 616–630.

[19] L. Jiao, L. Bo, and L. Wang, "Fast sparse approximation for least squares support vector machine," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 685–697, May 2007.

[20] O. Koyejo, S. Acharyya, and J. Ghosh, "Retargeted matrix factorization for collaborative filtering," in *Proc. ACM Recommender Syst. Conf. (RecSys)*, 2013, pp. 49–56.

[21] F. De la Torre, "A least-squares framework for component analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1041–1055, Jun. 2012.

[22] Q. Le, T. Sarlós, and A. Smola, "Fastfood—Approximating kernel expansions in loglinear time," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2013, pp. 1–11.

[23] F. Li and Y. Yang, "A loss function analysis for classification methods in text categorization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 472–479.

[24] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Feb. 2008.

[25] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Inf. Retr.*, vol. 3, no. 2, pp. 127–163, 2000.

[26] T. K. Paul and T. Ogunfunmi, "Study of the convergence behavior of the complex kernel least mean square algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 9, pp. 1349–1363, Sep. 2013.

[27] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2007, pp. 1177–1184.

[28] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.

[29] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *J. Roy. Statist. Soc. B, Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.

[30] S. Xiang, F. Nie, G. Meng, C. Pan, and C. Zhang, "Discriminative least squares regression for multiclass classification and feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 11, pp. 1738–1754, Nov. 2012.

[31] S.-H. Yang and B.-G. Hu, "A stagewise least square loss function for classification," in *Proc. SIAM Int. Conf. Data Mining (SDM)*, 2008, pp. 120–131.