Discriminative Least Squares Regression for Multiclass Classification and Feature Selection

Shiming Xiang, Feiping Nie, Gaofeng Meng, Chunhong Pan, and Changshui Zhang, *Member, IEEE*

***Abstract*— This paper presents a framework of discriminative least squares regression (LSR) for multiclass classification and feature selection. The core idea is to enlarge the distance between different classes under the conceptual framework of LSR. First, a technique called ε-dragging is introduced to force the regression targets of different classes moving along opposite directions such that the distances between classes can be enlarged. Then, the ε-draggings are integrated into the LSR model for multiclass classification. Our learning framework, referred to as discriminative LSR, has a compact model form, where there is no need to train two-class machines that are independent of each other. With its compact form, this model can be naturally extended for feature selection. This goal is achieved in terms of $L_{2,1}$ norm of matrix, generating a sparse learning model for feature selection. The model for multiclass classification and its extension for feature selection are finally solved elegantly and efficiently. Experimental evaluation over a range of benchmark datasets indicates the validity of our method.**

***Index Terms*— Feature selection, least squares regression, multiclass classification, sparse learning.**

I. Introduction

LEAST SQUARES REGRESSION (LSR) is a widely-used statistical analysis technique. It has been adapted to many real-world situations. LSR earns its place as a fundamental tool due to its effectiveness for data analysis as well as its completeness in statistics theory. Many variants have been developed, including weighted LSR [1], partial LSR [2], and other extensions (for example, ridge regression [3]). In addition, the utility of LSR has been demonstrated in many machine learning problems, such as discriminative learning, manifold learning, clustering, semi-supervised learning, multitask learning, multiview learning, multilabel classification, and so on.

Linear regression was the first type of regression analysis to be strictly studied. Given a data set $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^m$ and a destination set $\{\mathbf{y}_i\}_{i=1}^n \subset \mathbb{R}^c$, where $\mathbf{y}_i$ is the image vector of $\mathbf{x}_i$, the popularly-used regularization for linear regression can be addressed as an optimization problem

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

where $\mathbf{W} \in \mathbb{R}^{m\times c}$ and $\mathbf{b} \in \mathbb{R}^c$ are to be estimated and λ is a regularization parameter, $||\cdot||_2$ denotes the L_2 norm, and $||\cdot||_F$ stands for the Frobenius norm of matrix.

In data analysis, (1) is often applied to data fitting where each $\mathbf{y}_i$ is a continuous observation. When it is employed for data classification, $\mathbf{y}_i$ is manually assigned as "+1/−1" for two-class problems or a class label vector for multiclass problems. For classification tasks, it is desired that, geometrically, the distances between data points in different classes are as large as possible after they are transformed. The motivation behind this criterion is very similar to those used for distance measure learning [4], [5]. But, formally, (1) does not contain the information related to such a geometrical criterion.

To enlarge the distance between classes, Leski proposed a LSR model via the squared approximation of the misclassification errors [6]. However, the algorithm is designed for two-class classification problems. When it is considered with one-versus-rest training rule for multiclass extensions, formally one can obtain multiple models that are independent of each other. In other words, this will not generate an economical training framework for multiclass classification, and will not lead to a compact model that can be easily extended for other learning tasks.

The above observation motivates us to consider how to explicitly embed class label information into the LSR framework so that the distances between different classes can be enlarged. Under this motivation, our goal is to develop a LSR model with compact form for multiclass classification that can be naturally extended for multiclass feature selection.

In pattern recognition and machine learning, feature selection has been identified as a key component in developing robust algorithms for classification. Generally speaking, the goal of feature selection is to choose from the input data a subset of relevant features, with which better performance (high accuracy and low training time) of the learning machine could be achieved. In addition, feature selection is one of the most important approaches to dealing with high-dimensional data. The efforts in feature selection have surged in a few

Manuscript received September 30, 2011; revised July 16, 2012; accepted August 3, 2012. Date of publication September 11, 2012; date of current version October 15, 2012. This work was supported in part by the National Basic Research Program of China under Grant 2012CB316304, and the National Natural Science Foundation of China under Grant 61272331, Grant 61175025, Grant 61005036, and Grant 60835002.

S. Xiang, G. Meng, and C. Pan are with the National Laboratory of Pattern Recognition, Institute of Automation, Academy of Sciences, Beijing 100190, China (e-mail: smxiang@gmail.com; gfmeng@nlpr.ia.ac.cn; chpan@nlpr.ia.ac.cn).

F. Nie is with the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019 USA (e-mail: feipingnie@gmail.com)

C. Zhang is with the State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: zcs@mail.tsinghua.edu.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.2012.2212721

decades, with the development of numerous approaches and proposals for real world applications [7], [8]. In spite of many thoughtful attempts, it is still a challenging task in pattern recognition and machine learning.

Taking the class labels of data points into account or not, feature selection algorithms will fall into three categories: unsupervised algorithms [9], semi-supervised algorithms [10]–[12], and supervised algorithms [7], [13]. In addition, multiple criteria have been proposed, including maximum mutual information [14]–[16], maximum margin [11], minimum reconstruction errors [17], [18], low misclassification error in neural network [19], probabilistic prediction [20], sparse representation [21]–[28], stability [29], eigenvalue sensitivity [30], and so on.

The family of supervised feature selection algorithms can be divided into filters [14], [15], [31], wrappers [32], [33], and embedded methods [21], [23], [34], according to whether a specific classifier is integrated into the learning framework [7]. Filters evaluate the correlation or relevance of a feature with respect to the class label distribution of the data. Thus, they are developed independently of specific classifiers. Wrappers use the prediction with a given learning machine to score the relative usefulness of subsets of features. But the computational complexity is usually very high. In contrast, embedded methods incorporate feature selection as part of the training process [7]. As embedded methods are coupled with specific classifier, they often show good performance.

In the literature, regularization formulation with sparse representation has also been applied to feature selection. Motivated by the lasso [35], a widely used trick is to reformulate the support vector machine (SVM) with L_1 or L_0 norm [21], [22], [24]. For example, Bradley and Mangasarian constructed a L_1-SVM that typically yields sparse solution [36]. Note that, with L_1-SVM, the number of selected features is upper bounded by the number of training samples. To remedy this drawback, Wang *et al.* proposed a hybrid Huberized SVM with a combination of L_1 and L_2 norms [24]. However, most algorithms with sparse representation mainly focus on two-class classification problems. Additionally, they often require complex optimization procedures. Cawley *et al.* developed sparse multinomial logistic regression via Bayesian L_1 regularization [23]. Their algorithm can deal with multiclass feature selection. Many experiments have demonstrated that it is a powerful feature selection algorithm [23]. However, it can be computationally demanding in case of high-dimensional data. Actually, developing an efficient feature selection model along the line of sparse representation for multiclass classification problems is still a fundamental topic to be further studied.

Our goal is to construct a new LSR model for multiclass classification. The core idea is to embed class label information into the LSR formulation such that the distances between classes can be enlarged. In order to implement this idea, a technique called ε-dragging is introduced to force the regression targets of different classes moving along with opposite directions. Intrinsically, such a geometrical treatment follows the one-versus-rest training rule for multiclass problem. By the use of Hadamard product matrices, we describe the ε-dragging technique in terms of a single compact target function. This term is then added into the LSR framework and a learning model for multiclass classification is finally constructed. Our new LSR framework, referred to as discriminative LSR (DLSR), can explicitly utilize label information. It can also be naturally extended for feature selection.

The most important properties of our new approach are as follows.

1) To develop the DLSR approach, Hadamard product of matrices is introduced to organize the ε-draggings. This treatment yields a compact model form, which translates well the one-versus-rest training rule for multiclass classification.
2) The DLSR formulation for multiclass classification will yield a convex problem. By performing variable decoupling, only a group of linear equations needs to be solved in each iteration. This task can be further avoided via variable substitution. Thus, each iteration has only linear time complexity. The low time complexity will facilitate its real-world applications.
3) The DLSR formulation for multiclass classification can be naturally extended to develop a learning model for feature selection. This is achieved by forcing the $L_{2,1}$ norms on both the LSR term and the regularization term, generating a convex sparse learning problem. With theoretical analysis about this problem, it is divided into two convex subproblems, each of which can be efficiently solved.
4) Besides the theoretical guarantee about our DLSR formulation for classification, experiments on public benchmark datasets from many different fields indicate that the algorithm is comparable to classical algorithms, including the traditional LSR, logistic regression [37], LDA [38] and SVM [39].
5) The validity of the extension algorithm for feature selection is tested on a total of 20 data sets coming from different fields. It is also widely compared with the classical feature selection algorithms. Its validity is illustrated in high classification accuracy, convergence to global optimum, and effectiveness for high-dimensional data.

The remainder of this paper is organized as follows. In Section II, we present the DLSR approach for multiclass classification. In Section III, we extend our approach for feature selection. In Section IV, we report the experiments on multiclass classification. Experiments on feature selection are given in Section V. Conclusions are drawn in Section VI.

II. DLSR for Multiclass Classification

A. Problem Formulation and Learning Model

Given n training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ falling into c (≥ 2) classes, where $\mathbf{x}_i$ is a data point in $\mathbb{R}^m$ and $y_i \in \{1, 2, \ldots, c\}$ is the class label of $\mathbf{x}_i$, our goal is to develop a LSR framework such that the distances between classes are as large as possible.

One way to achieve our goal is to apply Leski's two-class LSR model [6] with the one-versus-rest training rule. This approach yields n independent subproblems where the subproblems have to be solved subsequently one by one. Here, we will develop a unique compact model for multiclass cases.

Note that an arbitrary set of c independent vectors in $\mathbb{R}^c$ is capable of identifying c classes uniquely. Thus, we can take $0-1$ class label vectors as the regression targets for multiclass classification. That is, for the jth class, $j = 1, 2, \ldots, c$, we define $\mathbf{f}_j = [0, \ldots, 0, 1, 0, \ldots, 0]^T \in \mathbb{R}^c$ with only the jth element equal to one [in other words, this is actually achieved in way of dummy (or unary) encoding where the 1 is the dummy]. Now our goal is to learn a linear function

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} + \mathbf{t} \tag{2}$$

such that for n training samples we have

$$\mathbf{f}_{y_i} \approx \mathbf{W}^T\mathbf{x}_i + \mathbf{t}, \qquad i = 1, 2, \ldots, n \tag{3}$$

where $\mathbf{W}$ is a transformation matrix in $\mathbb{R}^{m\times c}$ and $\mathbf{t}$ is a translation vector in $\mathbb{R}^c$.

Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n\times m}$ collect the n data points, and $\mathbf{Y} = [\mathbf{f}_{y_1}, \mathbf{f}_{y_2}, \ldots, \mathbf{f}_{y_n}]^T \in \mathbb{R}^{n\times c}$ record their images. Then, the conditions in (3) can be rewritten as

$$\mathbf{XW} + \mathbf{e}_n\mathbf{t}^T \approx \mathbf{Y} \tag{4}$$

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

We see in the jth column in $\mathbf{Y}$, only the elements corresponding to the data in the jth class are equal to one and all the remaining elements are zero. Thus, each column vector of $\mathbf{Y}$ actually stipulates a type of binary regression with target "+1" for the jth class and target "0" for the remaining classes. Although it is impossible for us to write out similar constraints used in two-class cases [6] with 0/1 outputs, we can drag these binary outputs far away along two opposite directions. That is, with a positive slack variable ε_i, we hope the output will become $1 + \varepsilon_i$ for the sample grouped into "1" and $-\varepsilon_i$ for the sample grouped into "0." In this way, the distance between two data points from different classes will be enlarged. This gears to the general criterion of enlarging the margin between classes for regression [39], [40].

Table I further explains the motivation, which reports six data points in three classes. Their class label vectors are listed in the third column. Now if we collect together the first component of the class label vectors, we can get "1, 1, 0, 0, 0, 0." This gives a binary-class partition, in which the first two points are divided into one class while the last four data points are divided into another class. After ε-draggings are performed, their images will be changed from "1, 1, 0, 0, 0, 0" to "$1+\varepsilon_{11}$,$1+\varepsilon_{21}$, $-\varepsilon_{31}$, $-\varepsilon_{41}$, $-\varepsilon_{51}$, $-\varepsilon_{61}$." As all εs are nonnegative, this treatment could help to enlarge the distance between classes after the data points are mapped.

To develop a unique compact optimization model for multiclass cases, we consider c columns together. Let $\mathbf{B} \in \mathbb{R}^{n\times c}$ be a constant matrix, in which the ith row and jth column element B_{ij} is defined as

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

TABLE I

PERFORMANCE OF ε-DRAGGING ON DATA POINTS IN THREE CLASSES

	class	$\mathbf{y}$	$\mathbf{y}$ after ε-dragging	constraint
$\mathbf{x}_1$	1	[1, 0, 0]	$[1+\varepsilon_{11}, -\varepsilon_{12}, -\varepsilon_{13}]$	$\varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13} \geq 0$
$\mathbf{x}_2$	1	[1, 0, 0]	$[1+\varepsilon_{21}, -\varepsilon_{22}, -\varepsilon_{23}]$	$\varepsilon_{21}, \varepsilon_{22}, \varepsilon_{23} \geq 0$
$\mathbf{x}_3$	2	[0, 1, 0]	$[-\varepsilon_{31}, 1+\varepsilon_{32}, -\varepsilon_{33}]$	$\varepsilon_{31}, \varepsilon_{32}, \varepsilon_{33} \geq 0$
$\mathbf{x}_4$	2	[0, 1, 0]	$[-\varepsilon_{41}, 1+\varepsilon_{42}, -\varepsilon_{43}]$	$\varepsilon_{41}, \varepsilon_{42}, \varepsilon_{43} \geq 0$
$\mathbf{x}_5$	3	[0, 0, 1]	$[-\varepsilon_{51}, -\varepsilon_{52}, 1+\varepsilon_{53}]$	$\varepsilon_{51}, \varepsilon_{52}, \varepsilon_{53} \geq 0$
$\mathbf{x}_6$	3	[0, 0, 1]	$[-\varepsilon_{61}, -\varepsilon_{62}, 1+\varepsilon_{63}]$	$\varepsilon_{61}, \varepsilon_{62}, \varepsilon_{63} \geq 0$

Geometrically, each element in $\mathbf{B}$ corresponds to a dragging direction. That is, "+1" means it points to the positive axis, while "-1" means it points to the negative axis. Performing the above ε-dragging on each element of $\mathbf{Y}$ and recording these $\{\varepsilon\}$ by matrix $\mathbf{M} \in \mathbb{R}^{n\times c}$, we have the following residual:

$$\mathbf{XW} + \mathbf{e}_n\mathbf{t}^T - (\mathbf{Y} + \mathbf{B} \odot \mathbf{M}) \approx \mathbf{0} \tag{6}$$

where $\odot$ is a Hadamard product operator of matrices.

Now following the regularized LSR framework, we can obtain a learning model as follows:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{t},\mathbf{M}} \quad & \left|\left|\mathbf{XW} + \mathbf{e}_n\mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}\right|\right|_F^2 + \lambda||\mathbf{W}||_F^2 \\ \text{s.t.} \quad & \mathbf{M} \geq \mathbf{0} \end{aligned} \tag{7}$$

where λ is a positive regularization parameter.

Application of this new notation to (1) yields

$$\min_{\mathbf{W},\mathbf{t}} \left|\left|\mathbf{XW} + \mathbf{e}_n\mathbf{t}^T - \mathbf{Y}\right|\right|_F^2 + \lambda||\mathbf{W}||_F^2. \tag{8}$$

In contrast to (8), we add a term $\mathbf{B} \odot \mathbf{M}$ in (7), which is related to ε-draggings to enlarge the distances between different classes. Accordingly, the learning model now turns out to be a constrained optimization problem.

With our formulation, we see the c subproblems are grouped together to share a unique learning model. This will yield two advantages. One is that, even with the similar one-versus-rest training rule, we only need to solve them once for multiclass cases. Another advantage is that the model has a compact form, which can be further refined in order to implement feature selection based on sparse representation.

B. Solving the Optimization Model

Based on convex optimization theory, it can be easily justified that problem (7) is convex, which has a unique optimal solution. Here, an iterative method will be presented. The optimization of (7) with respect to $\mathbf{W}$ and $\mathbf{t}$ is based on the following theorem [41].

Theorem 1: Given $\mathbf{M}$, and let $\mathbf{R} = \mathbf{Y} + \mathbf{B} \odot \mathbf{M} \in \mathbb{R}^{n\times c}$. Then, the optimal $\mathbf{W}$ and $\mathbf{t}$ in (7) can be calculated as

$$\mathbf{W} = (\mathbf{X}^T\mathbf{HX} + \lambda\mathbf{I}_m)^{-1}\mathbf{X}^T\mathbf{HR} \tag{9}$$

and

$$\mathbf{t} = \frac{(\mathbf{R}^T\mathbf{e}_n - \mathbf{W}^T\mathbf{X}^T\mathbf{e}_n)}{n} \tag{10}$$

where $\mathbf{I}_m$ is a $m \times m$ identity matrix, and $\mathbf{H} = \mathbf{I}_n - (1/n)\mathbf{e}_n\mathbf{e}_n^T$, in which $\mathbf{I}_n$ is a $n \times n$ identity matrix.

Proof: Based on (7), we denote $g(\mathbf{W}, \mathbf{t}) = ||\mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{t}^T - \mathbf{R}||_F^2 + \lambda||\mathbf{W}||_F^2$. According to matrix theory, we have

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

Furthermore, we have

$$\frac{\partial g(\mathbf{W}, \mathbf{t})}{\partial \mathbf{W}} = \mathbf{0}$$
$$\Rightarrow \mathbf{X}^T\left(\mathbf{X}\mathbf{W} + \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T\mathbf{R} - \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T\mathbf{X}\mathbf{W} - \mathbf{R}\right) + \lambda\mathbf{W} = \mathbf{0}$$
$$\Rightarrow \mathbf{X}^T\left(\mathbf{I}_n - \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T\right)\mathbf{X}\mathbf{W} - \mathbf{X}^T\left(\mathbf{I}_n - \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T\right)\mathbf{R} + \lambda\mathbf{W} = \mathbf{0}$$
$$\Rightarrow \mathbf{W} = (\mathbf{X}^T\mathbf{H}\mathbf{X} + \lambda\mathbf{I}_m)^{-1}\mathbf{X}^T\mathbf{H}\mathbf{R}.$$

Thus, we finish the proof. ■

Now we consider the optimization with respect to $\mathbf{M} \in \mathbb{R}^{n\times c}$. Given $\mathbf{W}$ and $\mathbf{t}$, and let $\mathbf{P} = \mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{t}^T - \mathbf{Y}$ record the regression errors of the n data points. Then, $\mathbf{M}$ can be solved from the following optimization problem:

$$\min_{\mathbf{M}} \quad ||\mathbf{P} - \mathbf{B} \odot \mathbf{M}||_F^2, \quad \text{s.t.} \quad \mathbf{M} \geq \mathbf{0}. \tag{11}$$

Based on the fact that the squared Frobenius norm of matrix can be decoupled element by element, (11) can be decoupled equivalently into $n \times c$ subproblems. For the ith row and jth column element M_{ij}, we have

$$\min_{M_{ij}} \quad (P_{ij} - B_{ij}M_{ij})^2, \quad \text{s.t.} \quad M_{ij} \geq 0 \tag{12}$$

where P_{ij} and B_{ij} are the ith row and jth elements of $\mathbf{P}$ and $\mathbf{B}$, respectively. Now we have the following theorem.

Theorem 2: The optimal solution to (12) is

$$M_{ij} = \max(B_{ij}P_{ij}, 0). \tag{13}$$

Note that $B_{ij}^2 = 1$. Thus, we have $(P_{ij} - B_{ij}M_{ij})^2 = (B_{ij}P_{ij} - M_{ij})^2$. Considering the nonnegative constraint about M_{ij}, we can get (13). Accordingly, $\mathbf{M}$ in (11) can be finally calculated as follows:

$$\mathbf{M} = \max(\mathbf{B} \odot \mathbf{P}, \mathbf{0}). \tag{14}$$

C. Algorithm of DLSR

On the basis of Theorems 1 and 2, we develop an iterative method that solves the primal problem. Algorithm 1 lists the steps of the DLSR algorithm.

In step 1, $\mathbf{W}_0$ and $\mathbf{t}_0$ are allocated to store the results obtained in the previous iteration. In step 4, matrix $\mathbf{U}$ is calculated in advance. With this variable substitution, in each iteration, $\mathbf{W}$ in (9) can be obtained via step 8.

To analyze the convergence of Algorithm 1, we denote the objective function in (7) by $G(\mathbf{W}, \mathbf{t}, \mathbf{M})$. Then, we have the following theorem.

Theorem 3: The Algorithm 1 monotonically decreases the value of $G(\mathbf{W}, \mathbf{t}, \mathbf{M})$.

Algorithm 1 *DLSR*

Input: n data points $\{\mathbf{x}_i\}_{i=1}^n$ in $\mathbb{R}^m$, and their corresponding class labels $\{y_i\}_{i=1}^n \subset \{1, 2, \ldots, c\}$; parameter λ in (7); and maximum number of iterations T.

```
1: Allocate $\mathbf{M}$, $\mathbf{W}$, $\mathbf{W}_0$, $\mathbf{t}$, and $\mathbf{t}_0$.
2: $\mathbf{M} = \mathbf{0}$, $\mathbf{W}_0 = \mathbf{0}$, and $\mathbf{t}_0 = \mathbf{0}$.
3: Construct $\mathbf{X}$ and $\mathbf{Y}$ in (4), and $\mathbf{B}$ according to (5).
4: Let $\mathbf{U} = (\mathbf{X}^T\mathbf{H}\mathbf{X} + \lambda\mathbf{I}_m)^{-1}\mathbf{X}^T\mathbf{H}$.
5: Let $k = 1$.
6: while $k < T$ do
7:    $\mathbf{R} = \mathbf{Y} + \mathbf{B} \odot \mathbf{M}$.
8:    $\mathbf{W} = \mathbf{U}\mathbf{R}$,   $\mathbf{t} = \frac{1}{n}\mathbf{R}^T\mathbf{e}_n - \frac{1}{n}\mathbf{W}^T\mathbf{X}^T\mathbf{e}_n$.
9:    $\mathbf{P} = \mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{t}^T - \mathbf{Y}$.
10:   $\mathbf{M} = \max(\mathbf{B} \odot \mathbf{P}, \mathbf{0})$.
11:   if $(||\mathbf{W} - \mathbf{W}_0||_F^2 + ||\mathbf{t} - \mathbf{t}_0||_2^2) < 10^{-4}$, then
12:      Stop.
13:   end if
14:   $\mathbf{W}_0 = \mathbf{W}$,   $\mathbf{t}_0 = \mathbf{t}$,   $k = k + 1$.
15: end while
16: Output $\mathbf{W}$ and $\mathbf{t}$.
```

Proof: Denote the value of the objective function at the $(t-1)$-th iteration by $G(\mathbf{W}^{t-1}, \mathbf{t}^{t-1}, \mathbf{M}^{t-1})$. During the tth iteration, we first fix $\mathbf{M}^{t-1}$ and solve subproblem $\min_{\mathbf{W},\mathbf{t}} G(\mathbf{W}, \mathbf{t}, \mathbf{M}^{t-1})$. Solving it via (9) and (10) yields the optimal solution $(\mathbf{W}^t, \mathbf{t}^t)$ at the tth iteration. Since this subproblem is convex, naturally we have

$$G(\mathbf{W}^{t-1}, \mathbf{t}^{t-1}, \mathbf{M}^{t-1}) \geq G(\mathbf{W}^t, \mathbf{t}^t, \mathbf{M}^{t-1}). \tag{15}$$

Next, we solve subproblem $\min_{\mathbf{M}\geq\mathbf{0}} G(\mathbf{W}^t, \mathbf{t}^t, \mathbf{M})$ by fixing $(\mathbf{W}^t, \mathbf{t}^t)$. Solving it via (14) yields the optimal $\mathbf{M}^t$. Due to the convexity of this subproblem, it follows:

$$G(\mathbf{W}^t, \mathbf{t}^t, \mathbf{M}^{t-1}) \geq G(\mathbf{W}^t, \mathbf{t}^t, \mathbf{M}^t). \tag{16}$$

Combining (15) and (16) together, we get

$$G(\mathbf{W}^{t-1}, \mathbf{t}^{t-1}, \mathbf{M}^{t-1}) \geq G(\mathbf{W}^t, \mathbf{t}^t, \mathbf{M}^t). \tag{17}$$

In this way, the above conclusion is proved. ■

Finally, we give some explanations about the above algorithm. First, we point out that the elements (the quantities of ε-draggings) of the optimum $\mathbf{M}$ are all finite. As can be witnessed from (13), the elements of $\mathbf{M}$ will be dropped into the interval of zero and the maximum absolute element of $\mathbf{P}$. Note that $\mathbf{P}$ collects the regression errors of the training samples. These errors are all finite since the optimization is started at finite cost function values. Thus, it is unnecessary to regularize $\mathbf{M}$ to avoid εs growing to infinity.

Second, we use an example to explain the performance of ε-draggings. Fig. 1(a) shows 60 training samples in $\mathbb{R}^2$ space, which belong to three classes. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{60}] \in \mathbb{R}^{2\times 60}$ collect these data points sequentially class by class (for example, the first 20 data points in $\mathbf{X}$ belong to the first class). Now we employ $\mathbf{X}$ to train the model in (7) with ε-draggings and that in (8) without ε-draggings. Correspondingly, we got two optimal transformations: $\mathbf{y} = \mathbf{W}^T\mathbf{x} + \mathbf{t}$ from (7) and $\mathbf{y} = \mathbf{W}_0^T\mathbf{x} + \mathbf{t}_0$ from (8). Let the images of data points in $\mathbf{X}$ under these two transformations be $\mathbf{I}$ and $\mathbf{I}_0$, respectively. Actually we have $\mathbf{I} = [\mathbf{W}^T\mathbf{x}_1 + \mathbf{t}, \ldots, \mathbf{W}^T\mathbf{x}_{60} + \mathbf{t}]$ and $\mathbf{I}_0 = [\mathbf{W}_0^T\mathbf{x}_1 + \mathbf{t}_0, \ldots, \mathbf{W}_0^T\mathbf{x}_{60} + \mathbf{t}_0]$. Now the differences of the images caused by ε-draggings can be evaluated as $\mathbf{D} = \mathbf{I} - \mathbf{I}_0$ $(\in \mathbb{R}^{3\times 60})$. Each column of $\mathbf{D}$ includes three components of

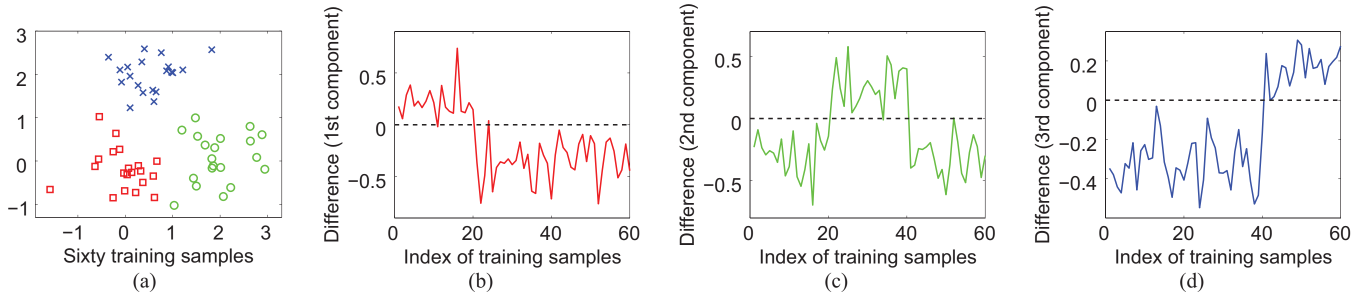

Fig. 1. Training samples and the differences of the images caused by ε-draggings. (a) 60 training samples in three classes. (b) First components of the differences caused by ε-draggings. (c) Second components of the differences caused by ε-draggings. (d) Third components of the differences caused by ε-draggings.

a difference vector. Fig. 1(b)–(d) shows the three components recorded, respectively, in the first, second, and third row of $\mathbf{D}$. We see that the three curves clearly show patterns. For example, according to the construction of the class label matrix $\mathbf{Y}$ in (4), the first 20 data points will be divided into one class, while the last 40 data points will be divided into another class. With ε-draggings, we hope the first 20 data points could be transformed above one, while the last 40 data points could be transformed below zero. In other words, the differences of the first 20 data points could be positive, while those of the last 40 ones could be negative. The differences shown in Fig. 1(b) demonstrate this fact. We see ε-dragging indeed helps to enlarge the distances of the classes. This fact can also be witnessed in Fig. 1(c) and (d).

III. Extending DLSR for Feature Selection

A. Optimization Model

Suppose we are given n training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, which belong to c (≥ 2) classes. Here, $\mathbf{x}_i \in \mathbb{R}^m$ is a data point and $y_i \in \{1, 2, \ldots, c\}$ is its class label. Our goal is to select d features from the original m features for classification.

Straightforwardly, the task is to find a $0-1$ selection matrix $\mathbf{W} \in \{0, 1\}^{m \times d}$ such that $\tilde{\mathbf{x}} = \mathbf{W}^T\mathbf{x}$ ($\in \mathbb{R}^d$) is a sub-vector of $\mathbf{x}$, where each row and each column in $\mathbf{W}$ has only one component equal to 1. Directly finding such a $0-1$ selection matrix is proven to be a NP hard problem [42]. A commonly-used way is to consider the feature selection problem by taking $\mathbf{W}$ as a transformation matrix.

If some rows of $\mathbf{W}$ were equal to zero, the data dimensions that correspond to these rows could be removed, i.e., a selection of dimensions could be performed. We can formalize this as follows. Let $\mathbf{W}^i$ be the ith row of $\mathbf{W}$. Then, a new vector $\bar{\mathbf{w}}$, which collects the L_2 norms of the row vectors of $\mathbf{W}$, can be constructed as

$$\bar{\mathbf{w}} = [\, ||\mathbf{W}^1||_2, ||\mathbf{W}^2||_2, \ldots, ||\mathbf{W}^m||_2 \,]^T \in \mathbb{R}^m \tag{18}$$

where $||\mathbf{W}^i||_2 = \sqrt{\sum_{j=1}^{c} W_{ij}^2}$, $i = 1, 2, \ldots, m$. Now we see that constructing d nonzero rows in $\mathbf{W}$ is just equivalent to forcing the number of nonzero entities in $\bar{\mathbf{w}}$ equal to d

$$||\bar{\mathbf{w}}||_0 = d. \tag{19}$$

Unfortunately, directly solving the problem with constraints in L_0 norm is also a NP hard problem. Alternatively, we consider approximating L_0 norm with L_1 norm [43], [44] and use the following $L_{2,1}$ norm of matrix $\mathbf{W}$ to develop the learning model for feature selection

$$\|\mathbf{W}\|_{2,1} = \|\bar{\mathbf{w}}\|_1 = \sum_{i=1}^{m} \sqrt{\sum_{j=1}^{c} W_{ij}^2}. \tag{20}$$

Formally, $||\mathbf{W}||_F^2$ in (7) will be replaced by $||\mathbf{W}||_{2,1}$ for feature selection. Furthermore, we can also replace the least squares term in (7) by $||\cdot||_{2,1}$. That is, $||\mathbf{A}||_F^2$ will be replaced by $||\mathbf{A}||_{2,1}$, where $\mathbf{A} = \mathbf{XW} + \mathbf{e}_n\mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}$. This treatment is motivated as follows. First, in contrast to $||\mathbf{A}||_F^2$, $||\mathbf{A}||_{2,1}$ will be more robust to outliers. This is due to the fact that the residuals are not squared and thus outliers have less importance. Second, similarly to (18), we can construct a vector $\bar{\mathbf{a}}$ in $\mathbb{R}^n$ from $\mathbf{A}$, and force some entities in $\bar{\mathbf{a}}$ to be zero. Then, optimizing a sparse $\bar{\mathbf{a}}$ is just equivalent to minimizing the $||\mathbf{A}||_{2,1}$. This treatment could result in some zero rows in $\mathbf{A}$, which will facilitate the learning for the optimal transformation. Note that L_1 norm of $\mathbf{A}$ can also be employed to enhance the robustness of the model. However, $L_{2,1}$ norm-based loss function is rotational invariant [45], and will lead to unchanged loss value under orthogonal transformations. But L_1 norm-based one does not have this desirable property.

Now (7) can be extended for feature selection as follows:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{t},\mathbf{M}} \quad & ||\mathbf{XW} + \mathbf{e}_n\mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}||_{2,1} + \lambda ||\mathbf{W}||_{2,1} \\ \text{s.t.} \quad & \mathbf{M} \geq \mathbf{0} \end{aligned} \tag{21}$$

where λ is a positive tradeoff parameter.

Note that $L_{2,1}$ is a norm. Thus, $||\cdot||_{2,1}$ is a convex function. Additionally, the constraint $\mathbf{M} \geq \mathbf{0}$ is also convex. Then, (21) is convex and has therefore only one global minimum.

B. Solving the Optimization Model With Given $\mathbf{M}$

Note that problem (21) is a nonlinear optimization problem. Here, we develop an iterative method to solve it. Like for problem (7), the first step is to solve $\mathbf{W}$ and $\mathbf{t}$ by fixing matrix $\mathbf{M}$. Let $\mathbf{T} = \mathbf{Y} + \mathbf{B} \odot \mathbf{M} \in \mathbb{R}^{n \times c}$. Then, (21) will turn out to be the following subproblem:

$$\min_{\mathbf{W},\mathbf{t}} \quad ||\mathbf{XW} + \mathbf{e}_n\mathbf{t}^T - \mathbf{T}||_{2,1} + \lambda ||\mathbf{W}||_{2,1}. \tag{22}$$

In (22), there does not exist analytical solution for $\mathbf{W}$ and $\mathbf{t}$. A straightforward way to minimize it is the gradient descent approach, which requires an appropriate choice of the initial solution and a proper control of the step size in each iteration. To avoid these problems, here we reformulate (22) and develop an equivalent model easy for optimization.

In order to derive our optimization approach, we reformulate the problem in terms of homogeneous coordinates. For $\mathbf{x}$, its homogeneous coordinate is defined as $\tilde{\mathbf{x}} = [\mathbf{x}, u]^T$, where u is a scalar number.

Let $\tilde{\mathbf{W}} = [\mathbf{W}^T, \mathbf{t}]^T \in \mathbb{R}^{(m+1)\times c}$. With homogeneous coordinate, the transformation in (2) can be rewritten as

$$\mathbf{y} = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}. \tag{23}$$

To simplify (22), first we give a more general conclusion about the formulation with homogeneous coordinates.

Lemma 1: For transformation $\mathbf{y} = \mathbf{W}^T\mathbf{x} + \mathbf{t}$ with $\mathbf{W}$ and $\mathbf{t}$ to be optimized, suppose $h(\mathbf{t}) \geq 0$ holds for any $\mathbf{t}$, and

$$\lim_{\mathbf{t}\to\mathbf{0}} h(\mathbf{t}) = 0. \tag{24}$$

Let $\{\mathbf{W}^*, \mathbf{t}^*\}$ be the optimal solution to the following optimization problem:

$$\min_{\mathbf{W},\mathbf{t}} f\left(\left[\mathbf{X},\ u\mathbf{e}_n\right]\begin{bmatrix}\mathbf{W}\\ \mathbf{t}^T\end{bmatrix}\right) + \lambda g(\mathbf{W}) + \lambda h(\mathbf{t}) \tag{25}$$

where $f(\cdot)$ and $g(\cdot)$ are two functions, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n\times m}$, $\mathbf{e}_n = [1, 1, \ldots, 1]^T \in \mathbb{R}^n$, and λ is a positive number.

Then, for $u \to \infty$, $\{\mathbf{W}^*, u\mathbf{t}^*\}$ will be the optimal solution to the following optimization problem:

$$\min_{\mathbf{W},\mathbf{t}} f(\mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{t}^T) + \lambda g(\mathbf{W}). \tag{26}$$

Proof: Here, a proof by contradiction is given as follows. Suppose $\{\mathbf{W}^*, u\mathbf{t}^*\}$ is not the optimal solution to (26). Then there exists another solution $\{\hat{\mathbf{W}}, \hat{\mathbf{t}}\}$ such that

$$f(\mathbf{X}\hat{\mathbf{W}} + \mathbf{e}_n\hat{\mathbf{t}}^T) + \lambda g(\hat{\mathbf{W}}) < f(\mathbf{X}\mathbf{W}^* + u\mathbf{e}_n\mathbf{t}^{*T}) + \lambda g(\mathbf{W}^*). \tag{27}$$

In the case of $u \to \infty$, we have

$$\begin{aligned} f&\left(\left[\mathbf{X},\ u\mathbf{e}_n\right]\begin{bmatrix}\hat{\mathbf{W}}\\ \frac{1}{u}\hat{\mathbf{t}}^T\end{bmatrix}\right) + \lambda g(\hat{\mathbf{W}}) + \lambda h\left(\frac{1}{u}\hat{\mathbf{t}}\right)\\ &= f(\mathbf{X}\hat{\mathbf{W}} + \mathbf{e}_n\hat{\mathbf{t}}^T) + \lambda g(\hat{\mathbf{W}})\\ &< f(\mathbf{X}\mathbf{W}^* + u\mathbf{e}_n\mathbf{t}^{*T}) + \lambda g(\mathbf{W}^*)\\ &\leq f\left(\left[\mathbf{X},\ u\mathbf{e}_n\right]\begin{bmatrix}\mathbf{W}^*\\ \mathbf{t}^{*T}\end{bmatrix}\right) + \lambda g(\mathbf{W}^*) + \lambda h(\mathbf{t}^{*T}) \end{aligned} \tag{28}$$

In (28), the first equality holds since $(1/u)\hat{\mathbf{t}} = \mathbf{0}$ when $u \to \infty$. In this case, we have $h((1/u)\hat{\mathbf{t}}) = 0$. In addition, the first inequality in (28) holds according to (27), and the second inequality holds due to the fact that $h(\mathbf{t}^{*T}) \geq 0$.

From (28), we see that $\{\mathbf{W}^*, \mathbf{t}^*\}$ is not the optimal solution to problem (25). This is just a contradiction. Due to this contradiction, we have proven our claim. ■

Now for (22), we have the following theorem.

Theorem 4: Let $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_n]^T \in \mathbb{R}^{n\times(m+1)}$ collect the n homogeneous coordinates. In the case of $u \to \infty$, solving (22) is equivalent to solving the following problem:

$$\min_{\tilde{\mathbf{W}}} \ ||\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}||_{2,1} + \lambda||\tilde{\mathbf{W}}||_{2,1}. \tag{29}$$

Proof: According to Lemma 1, we define $f(\mathbf{A}) = ||\mathbf{A} - \mathbf{T}||_{2,1}$, $g(\mathbf{W}) = ||\mathbf{W}||_{2,1}$ and $h(\mathbf{t}) = ||\mathbf{t}^T||_{2,1}$. Then we can rewrite the objective function in (22) as follows:

$$||\mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{t}^T - \mathbf{T}||_{2,1} + \lambda||\mathbf{W}||_{2,1} = f(\mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{t}^T) + \lambda g(\mathbf{W}). \tag{30}$$

Note that $||\tilde{\mathbf{W}}||_{2,1} = ||\mathbf{W}||_{2,1} + ||\mathbf{t}^T||_{2,1}$, $\tilde{\mathbf{X}} = [\mathbf{X}, u\mathbf{e}_n]$, and $\tilde{\mathbf{X}}\tilde{\mathbf{W}} = \mathbf{X}\mathbf{W} + u\mathbf{e}_n\mathbf{t}^T$. Hence, the objective in (29) can be reformulated as

$$||\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}||_{2,1} + \lambda||\tilde{\mathbf{W}}||_{2,1} = f(\tilde{\mathbf{X}}\tilde{\mathbf{W}}) + \lambda g(\mathbf{W}) + \lambda h(\mathbf{t}). \tag{31}$$

Note again that $\lim_{\mathbf{t}\to\mathbf{0}} h(\mathbf{t}) = \lim_{\mathbf{t}\to\mathbf{0}} ||\mathbf{t}^T||_{2,1} = 0$. Now we can take (30) and (31) as the objective functions in (26) and (25). According to Lemma 1, the solution of (29) is equivalent to the solution of (22) which proves our claim. ■

According to Lemma 1 and Theorem 4, if we get the optimal solution $\{\mathbf{W}^*, \mathbf{t}^*\}$ to (29), we will obtain the optimal solution $\{\mathbf{W}^*, u\mathbf{t}^*\}$ to (22).

Now we face the task of how to solve (29). Naturally, we can consider the derivative of the objective function with respect to $\tilde{\mathbf{W}}$, and take it to be zero. This will give us a method to solve the optimization problem.

First, we consider the derivative of the term $||\tilde{\mathbf{W}}||_{2,1}$. According to the definition of $L_{2,1}$ norm, the derivative about the element $\tilde{W}_{ij}$ can be calculated as

$$\frac{\partial||\tilde{\mathbf{W}}||_{2,1}}{\partial\tilde{W}_{ij}} = \tilde{W}_{ij}\left(\sum_{k=1}^{c}\tilde{W}_{ik}^2\right)^{-0.5} = \frac{\tilde{W}_{ij}}{||\mathbf{w}^i||_2} \tag{32}$$

where $\mathbf{w}^i$ is the ith row of $\tilde{\mathbf{W}}$. Then, it is easy to obtain

$$\frac{\partial||\tilde{\mathbf{W}}||_{2,1}}{\partial\tilde{\mathbf{W}}} = \Sigma\tilde{\mathbf{W}} \tag{33}$$

where Σ is a diagonal matrix in $\mathbb{R}^{(m+1)\times(m+1)}$ with the ith diagonal component Σ_{ii} equal to $1/||\mathbf{w}^i||_2$.

Next, we consider the derivative of the term $||\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}||_{2,1}$ with respect to $\tilde{\mathbf{W}}$. Actually, it can be viewed as a composite function, which is generated by $f(\tilde{\mathbf{U}})$ with $f(\mathbf{U}) = ||\mathbf{U}||_{2,1}$ and $\mathbf{U} = \tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}$. This derivative can be obtained according to the chain rule. Hence, by use of simple matrix operations on the basis of (33), we have

$$\frac{\partial||\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}||_{2,1}}{\partial\tilde{\mathbf{W}}} = \tilde{\mathbf{X}}^T\mathbf{D}(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) \tag{34}$$

where $\mathbf{D}$ is a diagonal matrix in $\mathbb{R}^{n\times n}$ with the ith diagonal component D_{ii} equal to $1/||\tilde{\mathbf{x}}_i^T\tilde{\mathbf{W}} - \mathbf{f}_{y_i}^T||_2$. Here, $\mathbf{f}_{y_i}$ is the ith column of matrix $\mathbf{Y}$ [see the notation in (4)].

Finally, we take the derivative of the objective function in (29) and set it to zero

$$\tilde{\mathbf{X}}^T\mathbf{D}(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) + \lambda\Sigma\tilde{\mathbf{W}} = \mathbf{0}. \tag{35}$$

Equation (35) indicates that $\tilde{\mathbf{W}}$ can be solved as

$$\tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^T\mathbf{D}\tilde{\mathbf{X}} + \lambda\Sigma)^{-1}\tilde{\mathbf{X}}^T\mathbf{D}\mathbf{T}. \tag{36}$$

In (36), the regularization term $\lambda\Sigma$ is embedded into the calculation of the inverse matrix to reduce the under-determined configurations in case of fewer samples than dimensions.

Note that both Σ and $\mathbf{D}$ are dependent on $\tilde{\mathbf{W}}$. Therefore, $\tilde{\mathbf{W}}$ can be iteratively determined by use of Σ and $\mathbf{D}$ from the previous optimization step. Denoting the result at the tth step by $\tilde{\mathbf{W}}_t$, at the next step, we have

$$\Sigma_{ii}^{(t)} = \frac{1}{||\mathbf{w}_t^i||_2}, \quad i = 1, 2, \ldots, m+1 \tag{37}$$

$$D_{ii}^{(t)} = \frac{1}{||\tilde{\mathbf{x}}_i^T \tilde{\mathbf{W}}_t - \mathbf{f}_{y_i}||_2}, \quad i = 1, 2, \ldots, n \tag{38}$$

and

$$\tilde{\mathbf{W}}_{t+1} = (\tilde{\mathbf{X}}^T \mathbf{D}_t \tilde{\mathbf{X}} + \lambda \Sigma_t)^{-1} \tilde{\mathbf{X}}^T \mathbf{D}_t \mathbf{T} \tag{39}$$

where $\mathbf{w}_t^i$ is the ith row of $\tilde{\mathbf{W}}_t$, $\Sigma_{ii}^{(t)}$ is the ith diagonal matrix of Σ_t, and $D_{ii}^{(t)}$ is the ith diagonal matrix of $\mathbf{D}_t$.

Note that (29) is a convex problem and therefore its global minimum is determined within the optimization. In this way, the optimization problem in (22) is also solved. The steps of the optimization procedure explained above are briefly given in Algorithm 2 (see Section III-D).

C. Solving the Optimization Model With Given W *and* t

Let $\mathbf{P} = \mathbf{X}\mathbf{W} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y} \in \mathbb{R}^{n \times c}$. Given $\mathbf{W}$ and $\mathbf{t}$, (21) can be simplified as follows:

$$\min_{\mathbf{M}} \ ||\mathbf{P} - \mathbf{B} \odot \mathbf{M}||_{2,1}, \quad \text{s.t.} \quad \mathbf{M} \geq \mathbf{0}. \tag{40}$$

For (40), we have the following theorem.

Theorem 5: The optimal $\mathbf{M}$ in (40) is calculated as

$$\mathbf{M} = \max(\mathbf{B} \odot \mathbf{P}, \mathbf{0}). \tag{41}$$

Here, we see (11) and (40) have the same solution though they have different objective functions. Actually, based on $L_{2,1}$ norm, (40) can be divided into n subproblems, each of which optimizes a row of $\mathbf{M}$. Specifically, let P_{ij}, B_{ij}, and M_{ij} be the ith row and jth components of $\mathbf{P}$, $\mathbf{B}$, and $\mathbf{M}$. Further let $\mathbf{m}^i$ be the ith row of $\mathbf{M}$. Then, $\mathbf{m}^i$ can be obtained from the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{m}^i} \ \sqrt{\textstyle\sum_{j=1}^{c} (P_{ij} - B_{ij} M_{ij})^2} \\ &\text{s.t.} \ \ M_{ij} \geq 0, \qquad j = 1, 2, \ldots, c. \end{aligned} \tag{42}$$

Note that functions $f(x) = \sqrt{x}$ and $g(x) = x$ have the same monotonicity for $x \geq 0$, and thus there exists a bijection between them. As a result, the solution of (42) is equivalent to the optimization of $\sum_{j=1}^{c} (B_{ij} P_{ij} - M_{ij})^2$ with respect to M_{ij}. This leads to c subproblems as formulated in (12). Thus, (11) and (40) have the same optimum.

D. Algorithm of DLSR for Feature Selection

After the optimal $\mathbf{W}$ is obtained from (21), d features can be selected from the m original features. This goal can be achieved by sorting the entities of $\bar{\mathbf{w}}$ in (18). The steps are described as follows.

First, we calculate the sum of the squared entities of each row in $\mathbf{W}$. Let $\bar{\mathbf{w}} = [w_1, w_2, \ldots, w_m]^T$ be a vector in $\mathbb{R}^m$. The ith entity of $\bar{\mathbf{w}}$ is computed as

$$w_i = \left(\sum_{j=1}^{c} W_{ij}^2 \right)^{0.5}, \quad i = 1, 2, \ldots, m. \tag{43}$$

Algorithm 2 *DLSR for Feature Selection (DLSR-FS)*

```
Input: $n$ data points $\{\mathbf{x}_i\}_{i=1}^n$ in $\mathbb{R}^m$, and their corresponding class labels $\{y_i\}_{i=1}^n$; number of features to be selected $d$; parameter $\lambda$ in (21); large positive number $u$; and maximum number of iterations $T$.
1: Allocate $\mathbf{M}$, $\mathbf{W}$, $\mathbf{W}_0$, $\mathbf{t}$, $\mathbf{t}_0$, $\tilde{\mathbf{W}}$, and $\tilde{\mathbf{W}}_0$.
2: $k = 1$, $\mathbf{M} = \mathbf{0}$, $\mathbf{W}_0 = \mathbf{0}$, $\mathbf{t}_0 = \mathbf{0}$, $\tilde{\mathbf{W}}_0 = \mathbf{0}$.
3: Construct $\mathbf{X}$ and $\mathbf{Y}$ in (4), $\mathbf{B}$ according to (5), and $\tilde{\mathbf{X}}$ in (29).
4: while $k < T$ do
5:   $\mathbf{T} = \mathbf{Y} + \mathbf{B} \odot \mathbf{M}$.
6:   $\Sigma = \mathbf{I}_{m+1}$, $\mathbf{D} = \mathbf{I}_n$, here $\mathbf{I}_{m+1}$ and $\mathbf{I}_n$ are identity matrices.
7:   $t = 1$.
8:   while $t < T$ do
9:     $\tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^T \mathbf{D} \tilde{\mathbf{X}} + \lambda \Sigma)^{-1} \tilde{\mathbf{X}}^T \mathbf{D} \mathbf{T}$.
10:    $\Sigma_{ii} = 1/||\tilde{\mathbf{w}}^i||_2$, $i = 1, 2, \ldots, m+1$.
11:    $D_{ii} = 1/||\tilde{\mathbf{x}}_i^T \tilde{\mathbf{W}} - \mathbf{y}_i^T||_2$, $i = 1, 2, \ldots, n$.
12:    if $||\tilde{\mathbf{W}} - \tilde{\mathbf{W}}_0||_F^2 < 10^{-4}$, then
13:      Output $\tilde{\mathbf{W}}$.
14:    end if
15:    $\tilde{\mathbf{W}}_0 = \tilde{\mathbf{W}}$, $t = t + 1$.
16:  end while
17:  Assign the first $m$ rows of $\tilde{\mathbf{W}}$ to $\mathbf{W}$.
18:  Assign the last row of $\tilde{\mathbf{W}}$ (after transposed) to $\mathbf{t}$.
19:  $\mathbf{t} = u\mathbf{t}$.
20:  $\mathbf{P} = \mathbf{X}\mathbf{W} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y}$.
21:  $\mathbf{M} = \max(\mathbf{B} \odot \mathbf{P}, \mathbf{0})$.
22:  if $(||\mathbf{W} - \mathbf{W}_0||_F^2 + ||\mathbf{t} - \mathbf{t}_0||_2^2) < 10^{-4}$, then
23:    Break.
24:  end if
25:  $\mathbf{W}_0 = \mathbf{W}$, $\mathbf{t}_0 = \mathbf{t}$, $k = k + 1$.
26: end while
27: Calculate $\bar{\mathbf{w}}$ according to (43).
28: Output the indices of the first $d$ largest entities of $\bar{\mathbf{w}}$.
```

Then, we select out the first d largest entities in $\bar{\mathbf{w}}$ (as the d largest nonsparse ones), and output their indices for selecting d features.

Algorithm 2 lists the steps of the algorithm of DLSR for feature selection (DLSR-FS). Steps 5–19 are computed to solve (22), while steps 20 and 21 are employed to solve (40). For both the outer while loop and the inner while loop, we set the maximum number of iterations as T. Actually, on most datasets, the convergence reaches within about ten iterations when solving (22). In addition, the outer while loop stops within about 20 iterations. In our implementation, we set T to be 30.

Finally, we point out that, in Algorithm 2, u is positively infinite, according to Lemma 1. In implementation, we use $u = 10\,000$. Note that parameter d will not affect the computations during training (steps 4∼26) as we need to rank all of the m source features. Thus, only the parameter λ has to be optimized with respect to the given data.

E. Algorithm Analysis

Here, we analyze the convergence of Algorithm 2. Denote the objective function in (21) by $F(\mathbf{W}, \mathbf{t}, \mathbf{M})$. Then we have the following theorem.

Theorem 6: Algorithm 2 monotonically decreases the value of $F(\mathbf{W}, \mathbf{t}, \mathbf{M})$ during iterations.

Proof: Denote the solution obtained at time $t-1$ by $(\mathbf{W}^{t-1}, \mathbf{t}^{t-1}, \mathbf{M}^{t-1})$. Note that (21) is convex, which is further divided into two convex subproblems, as formulated in (22) and (40), respectively. At the tth iteration, we first fix $\mathbf{M}^{t-1}$ and solve (22). After the optimization is fulfilled, we can get the optimal solution $(\mathbf{W}^t, \mathbf{t}^t)$. Thus for currently optimal point $(\mathbf{W}^{t-1}, \mathbf{t}^{t-1})$, we have

$$F(\mathbf{W}^{t-1}, \mathbf{t}^{t-1}, \mathbf{M}^{t-1}) \geq F(\mathbf{W}^t, \mathbf{t}^t, \mathbf{M}^{t-1}). \tag{44}$$

Next, (40) is solved by fixing $\mathbf{W}^{t-1}, \mathbf{t}^{t-1}$. In this way, we get its optimum $\mathbf{M}^t$ via (41). Therefore, for currently optimal point $\mathbf{M}^{t-1}$, we have

$$F(\mathbf{W}^t, \mathbf{t}^t, \mathbf{M}^{t-1}) \geq F(\mathbf{W}^t, \mathbf{t}^t, \mathbf{M}^t). \tag{45}$$

Combining (44) and (45) together, it follows:

$$F(\mathbf{W}^{t-1}, \mathbf{t}^{t-1}, \mathbf{M}^{t-1}) \geq F(\mathbf{W}^t, \mathbf{t}^t, \mathbf{M}^t). \tag{46}$$

In this way, we have proven our claim. ■

IV. Experimental Evaluation of DLSR

A. Data Sets

Vehicle: This data set, taken from the UCI machine learning repository,[1] contains 1186 data points with 18 dimensions. Each data point describes a silhouette of vehicle in image. The purpose is to classify a given silhouette as one of four types of vehicle.

AT and T Data Set: It includes 40 different persons, and each person has ten gray images with different expressions and facial details [46]. The size of each image is 28×23. Thus, the source dimensionality of data points is 644.

Umist Data Set: It contains the face images of 20 different persons.[2] The size of each image is 56×46. The source dimensionality is 2576.

AR Data Set: The face images of 120 individuals are used to construct a data set [47]. For each subject, seven images are randomly selected, including different facial expressions and illuminations. Each image is down-sampled to have 32×24 pixels. Thus, the source dimensionality is 768.

Usps Data Set: The images of ten digits are used in this paper.[3] For each digit, 200 images with 16×16 pixels are randomly selected to construct a data set. The source dimensionality is 256.

COIL-20 Data Set: It includes 20 objects [48], each of which has 72 gray images, which are taken from different view directions. Each image is down-sampled to have 16×16 pixels. Thus, the input dimensionality is 256.

Cora-OS Data Set: It is a subset containing the research papers about operating system (OS) [49].

WebKB Data Sets: WebKB-CL, WebKB-WT, and WebKB-WC. The data sets contain a subset consisting of about 3200 web pages from computer science departments of three schools (Cornell, Washington, and Wisconsin).[4]

Table II describes these data sets. For the last four data sets, principal component analysis is used to project them into 200-D subspace.[5]

TABLE II

Brief Description of the Data Sets for Classification

Data set	Total num.	Train. num.	Classes	Features
Vehicle	846	340	4	18
AT&T	400	160	40	644
Umist	575	240	20	2576
AR	840	360	120	768
Usps	2000	800	10	256
Coil20	1440	580	20	256
Cora-OS	1246	500	4	6737(200)
WebKB-CL	827	329	7	4134(200)
WebKB-WT	1166	469	7	4165(200)
WebKB-WC	1210	483	7	4189(200)

B. Parameter Settings and Comparisons

We will compare our algorithm, DLSR, with the classical LSR, LDA [38], linear SVM [39], and logistic regression [37]. For our algorithm, the optimum linear transformation in (2) is first learned via Algorithm 1. The images of the data points under this linear transformation are then employed for classification. The 1-NN classifier is finally used to fulfill this task. For LSR and LDA, the 1-NN classifier is also employed to classify the results mapped by linear transformation. Additionally, the one-versus-rest rule is employed in both SVM and logistic regression to fulfill the training tasks for multiclass classifications. Moreover, the results obtained with 1-NN classifier (baseline) by taking the original dimensions of data as input will also be reported for comparisons.

In our experiments, we take the maximum number of iterations $T = 30$ when running Algorithm 1. DLSR has only one parameter λ to be tested. We use ten-fold cross validation (but three-folds for AT&T) approach to select it for each data set. The candidate set is $\{0.0001, 0.001, 0.01, 0.1, 1.0, 10.0\}$. This set will be also used to select the regularization parameter λ in the classical LSR with the cross validation. For SVM, the multiclass classification task is solved via the LibSVM software [50]. Note that there exists an important regularization parameter C in SVM. We also use cross validation approach to select it from the candidate set $\{0.001, 0.01, 0.1, 1.0, 10.0, 100.0\}$. In addition, the classic LDA algorithm often suffers from the small sample size problem when dealing with high-dimensional data. In this case, the within-class scatter matrix may become singular. This will make LDA difficult to be performed. Thus, here the regularized LDA is performed with the regularization parameter γ that is tuned via cross validation from $\{0.0001, 0.001, 0.01, 0.1, 1.0, 10.0\}$. The selected parameters will be finally used in experiments.

Table III lists the recognition accuracy and the standard deviation, obtained with baseline, LDA, SVM, logistic regression, LSR, and our algorithm. These values are calculated from 20 random splits. Table II lists the number of training samples.

As can be seen from Table III, our algorithm achieves comparable mean classification accuracy on the data sets, compared with LDA. Actually, LDA can achieve satisfactory performances when the distribution of the data in each class

[1] Available at http://www.ics.uci.edu/mlearn/MLRepository.html.

[2] Available at http://images.ee.umist.ac.uk/danny/database.html.

[3] Available at http://www.kernel-machines.org/data.

[4] Available at http://www.cs.cmu.edu/∼webKB/.

[5] With the original high dimensions, logistic regression may fail to finish the training within 48 h.

TABLE III

CLASSIFICATION ACCURACY (%) OF THE DATA SETS LISTED IN TABLE II

Data set	Baseline	LDA	Linear SVM	Logistic	LSR	DLSR
Vehicle	68.5771±1.2983	65.8202±1.9684	70.0593±1.6441	**71.5163±1.3982**	65.8893±2.4089	68.9302±2.3801
AT&T	91.8125±2.3425	94.4035±1.6713	93.4792±2.1683	92.0208±1.7432	93.1042±1.3751	**94.4167±1.3813**
Umist	97.0746±1.3648	98.8207±0.7016	98.2537±0.9503	95.9403±1.6160	97.5672±1.2146	**98.8209±0.6949**
AR	42.1771±2.1970	93.2396±1.0014	63.0208±2.9798	80.3333±1.9289	90.6667±1.1057	**93.4271±0.7780**
Usps	94.0165±0.5932	91.1375±0.7985	**94.0167±0.6615**	92.6250±0.7568	91.0375±0.7176	93.0667±0.6681
Coil20	96.6395±0.6255	98.0930±0.5626	98.1279±0.5280	95.0698±0.9502	96.9826±0.6701	**98.5581±0.3780**
Cora-OS	48.8194±3.0273	65.8839±1.3760	64.9161±2.3626	68.2502±1.5190	65.8839±1.4613	**68.2645±1.3938**
WebKB-CL	59.6260±3.6190	69.8618±3.2790	71.3333±3.2435	68.3171±2.7123	71.2439±3.4999	**74.0163±2.6841**
WebKB-WT	76.1093±2.2639	81.9040±1.9850	81.2914±1.9788	78.2340±1.7332	83.9570±1.9885	**86.1755±1.7293**
WebKB-WC	60.5226±5.5134	73.1897±2.0815	75.7866±2.8435	76.3362±1.7818	75.3933±1.9910	**78.4321±2.1999**

TABLE IV

RESULTS OF PAIRED STUDENTS' t TESTS ON THE ">" RELATIONSHIP BETWEEN THE TWO ACCURACIES (MEANS) REPORTED IN TABLE III

	Vehicle	AT&T	Umist	AR	Usps	Coil20	Cora-OS	WebKB-CL	WebKB-WT	WebKB-WC
DLSR > Baseline	0	1	1	1	0	1	1	1	1	1
DLSR > LDA	1	0	0	1	1	1	1	1	1	1
DLSR > Linear SVM	0	1	1	1	0	1	1	1	1	1
DLSR > Logistic	0	1	1	1	1	1	1	1	1	1
DLSR > LSR	1	1	1	1	1	1	1	1	1	1

is Gaussian. When the data distributions are more complex than Gaussian, the performance of LDA is limited, which can be witnessed from the WebKB data sets.

The performance of our algorithm is comparable to that of SVM. From Table III, we can observe that it achieves higher accuracy on most data sets. In addition, it significantly outperforms SVM on the AR data set. In fact, there are 120 classes to be classified, and the classifiers are trained only with three samples in each class. This indicates that the classical SVM may fail to find the true decision function when the size of the classes is large and the size of the training set is very limited. This problem is attenuated to some degree with our DLSR that can be trained in a compact way for multiple classes.

The comparison of our algorithm and logistic regression reveals that our method provides superior performance in terms of classification mean accuracy on most data sets. As can be witnessed from the performances on Vehicle data set, logistic regression demonstrates its power in low-dimensional data space. However, on the high-dimensional data sets, for examples, AT&T, Umist, and AR data sets, our algorithm outperforms it significantly in accuracy. Compared with the classical LSR, Table III also indicates that the classification performance is significantly improved with our algorithm.

To illustrate the statistical difference between our approach and other algorithms, we did the paired student's t test on these data sets. Here, the hypothesis is "the classification (mean) accuracy obtained by DLSR is greater than that obtained by the other (given) method." Each test is run on two accuracy sequences, which are obtained from the 20 splits by our method and the given method. Table IV reports the results of the statistical tests. In each entity, "1" means that the hypothesis is correct (true) with probability 0.95, and "0" means that "the hypothesis is wrong (false)" with probability 0.95. For example, on the Vehicle data set (see Table III), the decision "68.9302 (Our) > 65.8202 (LDA)" is correct with probability 0.95. In summary, from Table IV, we see the decision that "our algorithm achieves higher classification accuracy" is correct on most data sets.

C. Computational Complexity of DLSR

In this section, we analyze the computational complexity of DLSR. In step 4 of Algorithm 1, the calculation of $\mathbf{X}^T\mathbf{H}\mathbf{X}$ will scale in $O(nm^2 + m)$ as $\mathbf{H}$ is just a centering matrix for $\mathbf{X}$. To calculate $\mathbf{V} = (\mathbf{X}^T\mathbf{H}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{U}$, we can solve the matrix equations via $\mathbf{U} = (\mathbf{X}^T\mathbf{H}\mathbf{X} + \lambda\mathbf{I})\mathbf{V}$. With matrix equations, the computational complexity will scale in about $O(nm^2)$. Thus, totally the complexity in step 4 is about $O(2nm^2 + m)$.

The steps of Algorithm 1 that are computationally most demanding are steps 9 and 10. On the basis of the variable substitution that is performed in step 4, the total computational complexity in each iteration will scale in about $O(2nmc)$, linearly in the number of training data points, the dimensionality of data, as well as the number of classes.

Finally, we point out that the computational cost of step 4 is quadratic in the dimensionality of data. To reduce the cost in the cases of $n < m$, we denote the centered data matrix by $\tilde{\mathbf{X}} = \mathbf{H}\mathbf{X} \in \mathbb{R}^{n\times m}$. The matrix $\mathbf{U}$ that is used in step 4 can be easily obtained from simple matrix operation

$$\mathbf{U} = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} + \lambda\mathbf{I}_m)^{-1}\tilde{\mathbf{X}}^T = \tilde{\mathbf{X}}^T(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T + \lambda\mathbf{I}_n)^{-1}. \qquad (47)$$

Similar to the above analysis, we now see that calculating $\mathbf{U}$ according to (47) will scale in about $O(2mn^2 + m)$. In the case of $n < m$, this reduces the computational complexity.

Table V reports the training time on the data sets. All the algorithms are performed on a PC with 2.83-GHz CPU

TABLE V

AVERAGE TRAINING TIME (SECONDS)

Data set	LDA	Linear SVM	Logistic	LSR	DLSR
Vehicle	0.012	0.015	0.037	0.001	0.015
AT&T	0.348	0.082	154.479	0.019	0.655
Umist	15.236	0.453	842.210	0.115	3.484
AR	1.399	0.552	1030.610	0.121	4.563
Usps	0.395	0.113	13.689	0.042	1.167
Coil20	0.239	0.150	20.806	0.030	0.988
Cora-OS	0.125	0.079	2.041	0.014	0.406
WebKB-CL	0.039	0.020	1.915	0.008	0.208
WebKB-WT	0.051	0.027	2.213	0.009	0.239
WebKB-WC	0.057	0.027	2.349	0.009	0.282

and 2.0-GB RAM, using MATLAB 7.0. The training time of our approach is approximately 10 times larger than SVM computational time and 30 times larger than classical LSR computational time. In contrast to logistic regression, the training time is significantly reduced. Compared with LDA, our approach need not perform the eigenvalue decomposition of matrix to find the eigenvectors corresponding to the specified number of largest eigenvalues. In addition, LDA needs to construct the within-class scatter matrix and the between-class scatter matrix. This will take much time when the number of classes is large and dimensions of data are high. For instance, the average training time on the Umist data set with LDA will take about 15.2 s whereas our algorithm only needs 3.5 s.

V. EXPERIMENTAL EVALUATION OF DLSR-FS

A. Data Sets

We evaluated our algorithm of DLSR for feature selection (DLSR-FS) on a total of 20 public data sets. Among them, the ten data sets listed in Table I are also included. In contrast, for datasets Cora-OS, WebKB-CL, WebKB-WT, and WebKB-WC, the original dimensions are now used to conduct experiments for feature selection. The details about the remaining ten data sets are described as follows (see Table VI).

One UCI Data Set: Protein.[6] This set contains 116 protein structure data points taken from six different classes.

Eight Microarray Data Sets in Bio-Informatics: CAR, Glioma, Lung, MLL, SRBCT, CLL-SUB-111, GLA-BRA-180, and TOX-171. The CAR dataset contains in total 174 samples in eleven classes from Affymetrix U95a GeneChips, while the Glioma dataset has 50 samples in four classes from Affymetrix U95av2 GeneChips [51]. The Lung data set totally has 203 samples in five classes [52]. The genes with standard deviations smaller than 50 expression units are removed, which produces a dataset with 3312 genes [53]. The MLL dataset contains 72 samples of mixed-lineage leukemia cancer data, which contains three leukemia classes. There are 5848 genes in each sample. The SRBCT dataset contains 83 samples in total. Each sample in this dataset contains 2308 gene expression values [54]. In addition, the details about the datasets of CLL-SUB-111 [55], GLA-BRA-180 [56], and TOX-171[7] can be found in Table IV.

TABLE VI

BRIEF DESCRIPTION OF THE DATA SETS FOR FEATURE SELECTION

Data set	Total num.	Train. num.	Classes	Features
Protein	116	48	6	20
Vehicle	846	340	4	18
AT&T	400	160	40	644
Umist	575	240	20	2576
AR	840	360	120	768
Usps	2000	800	10	256
Coil20	1440	580	20	256
Cora-OS	1246	500	4	6737
WebKB-CL	827	329	7	4134
WebKB-WT	1166	469	7	4165
WebKB-WC	1210	483	7	4189
CAR	174	66	11	9182
Glioma	50	20	4	4434
Lung	203	70	5	3312
MLL	72	30	3	5848
SRBCT	83	32	4	2308
CLL-SUB-111	111	41	3	11 340
GLA-BAR-180	180	72	4	49 151
TOX-171	171	68	4	5748
TDT2	600	240	4	36 771

Document Data Set: TDT2. It contains 600 samples,[8] which come from four classes.

B. Algorithms and Parameter Settings

We compared our algorithm with several typical feature selection algorithms in multiclass scenario, including mutual information (MI) [57], Gini coefficient (Gini) [58], student's T-test (T-test) [59], Fisher Score (FS) [31], ReliefF [8], [60], the minimum redundancy maximum relevance (mRMR) [14]. As the existing sparse SVM-based feature selection approaches are mainly designed for two-class classification tasks, we compared our algorithm with the sparse multinomial logistic regression via Bayesian L1 regularization (SBMLR) [23], which is proven to be a classical multiclass feature selection method developed via sparse learning. As a classical unsupervised feature selection approach, Laplacian score (LS) [9] will be also run for comparisons.

LS is a graph-based approach. Given training data points, five-nearest neighbor method is employed to construct the data graph. The affinity between two neighboring data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is evaluated as $w_{ij} = \exp(||\mathbf{x}_i - \mathbf{x}_j||_2^2/(2\sigma^2))$. In order to appropriately chose σ, we compute the mean value of the distances between all of the n training data points, and denote it by d_m. Based on this value, we construct a candidate set $\{d_m/8, d_m/4, d_m/2, d_m, 2d_m, 4d_m, 8d_m\}$, from which a σ is selected. Cross validation is performed to achieve this goal.

[6] Available at http://www.ics.uci.edu/ mlearn/MLRepository.html.

[7] Available at http://featureselection.asu.edu/datasets.php.

[8] Available at http://www.gabormelli.com/RKB/TDT-2_Benchmark_Task.

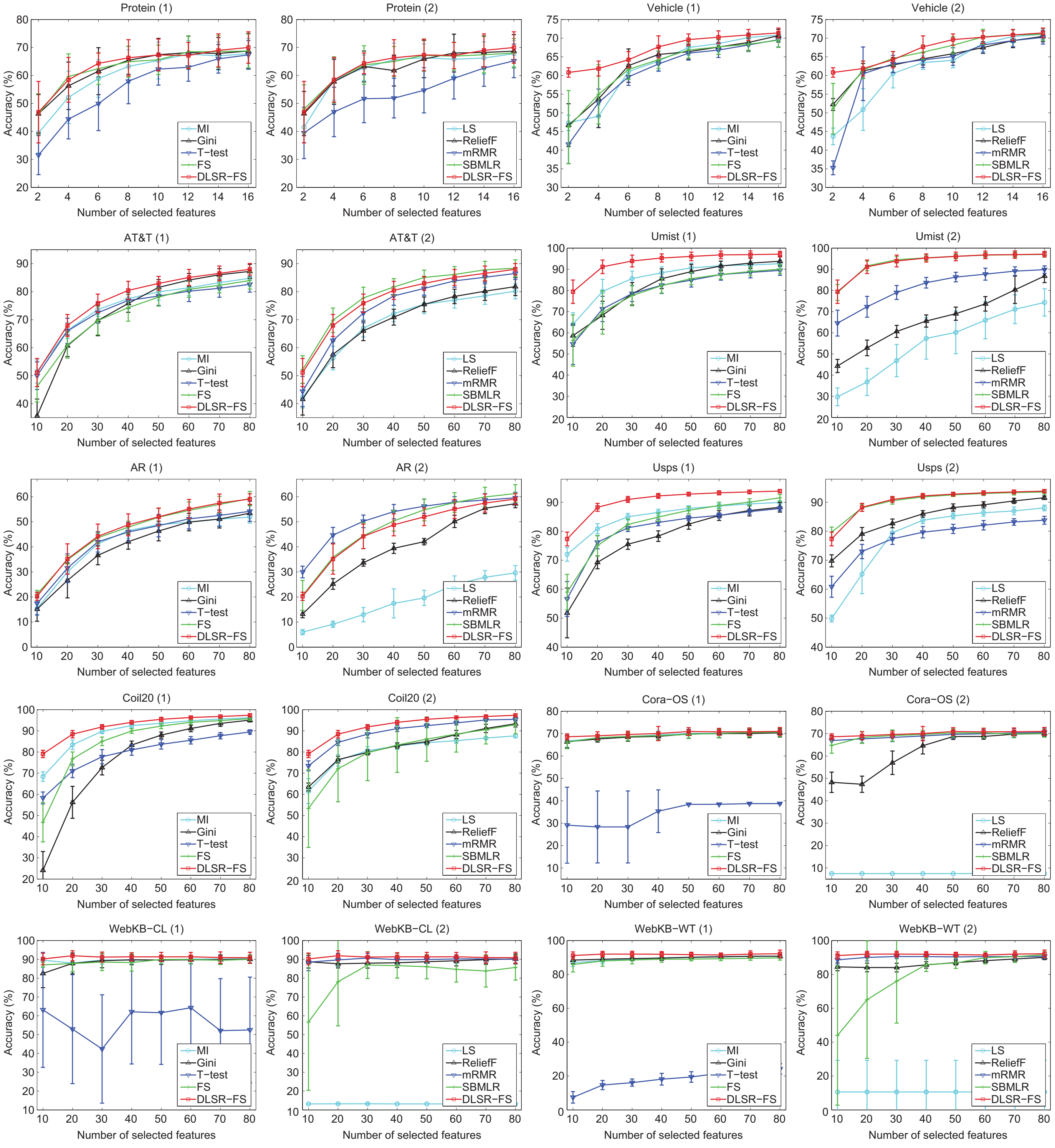

Fig. 2. Group I: The classification accuracy with different numbers of selected features. The final classification accuracy is calculated as the average of the 20 trials. In total, eight different numbers of selected features are evaluated, as indicated by the horizontal axis. To be clear, for each data set, two figures are employed to illustrate the accuracy and the standard deviation. Specifically, the first figure shows the results obtained by MI, Gini, T-test, FS, and our method, while the second figure illustrates those obtained by LS, ReliefF, mRMR, SBMLR, and our method.

Our algorithm has a parameter λ which should be tuned to data. We also use cross validation to select a proper λ. The candidate set for λ is $\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5\}$.

Classification accuracy is employed to evaluate the performance of feature selection. A linear SVM-classifier has been separately trained for each data set using the LibSVM [50]. In other words, given a training data set, the selected features will be employed to train a SVM classifier. Note that the regularization parameter C in SVM should be also tuned to data. We also use cross validation approach to select a

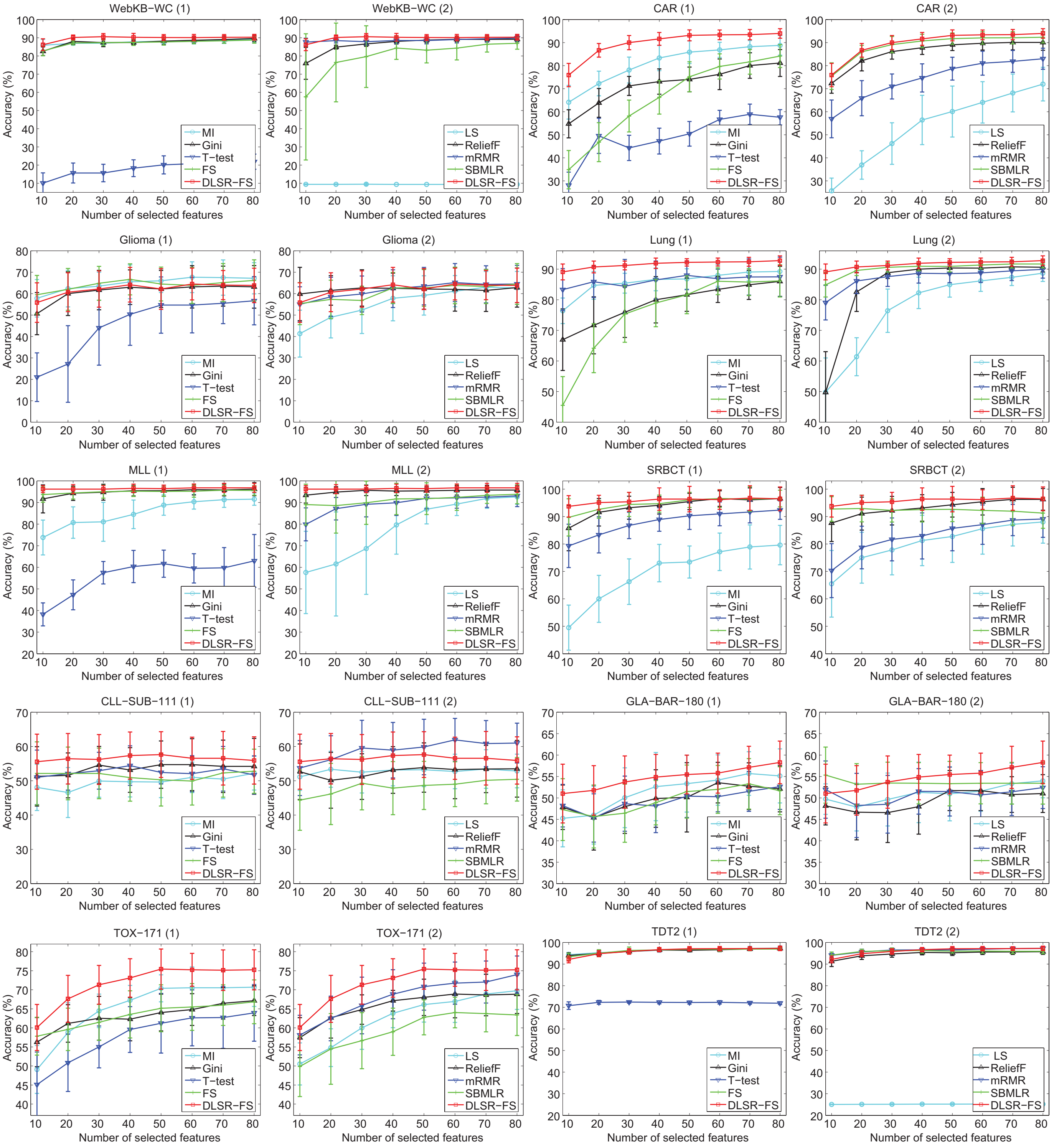

Fig. 3. Group II: The classification accuracy with different numbers of selected features. The final classification accuracy is calculated as the average of the 20 trials. In total, eight different numbers of selected features are evaluated, as indicated by the horizontal axis. To be clear, for each data set, two figures are employed to illustrate the accuracy and the standard deviation. Specifically, the first figure shows the results obtained by MI, Gini, T-test, FS, and our method, while the second figure illustrates those obtained by LS, ReliefF, mRMR, SBMLR, and our method.

proper C from the candidate set $\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$.

To perform the cross validation, the training set is first divided into a few subsets. Specifically, three-fold cross validation approach is performed on the ten data sets with number of training samples less than 200, while ten-fold cross validation approach is performed on the remaining ten data sets (see Table VI). Finally, cross validation approach is implemented for each group (λ, C). That is, for our algorithm, there are 56 groups of parameters λ and C to be selected. In experiments, a total of 20 trials are run for each group of parameters (λ, C). In each trial, about 40% data points are

TABLE VII

GROUP I: AVERAGE ACCURACY (%) AND THE STANDARD DEVIATION OF THE SELECTED TOP 80 FEATURES

	Protein	Vehicle	AT&T	Umist	AR	Usps
MI	67.7941±5.4379	70.8103±1.5283	84.7708±2.9632	92.6567±1.6217	51.9063±2.4476	89.9458±0.5562
Gini	68.7500±6.2914	70.6522±1.6958	87.2500±2.4941	93.8060±2.2793	53.4063±3.2333	88.2792±1.6680
T-test	67.3529±5.1077	69.6047±1.9675	82.6250±2.7774	89.4925±2.3296	54.1667±3.9851	87.8417±0.7803
FS	68.7647±6.0521	69.4565±1.9379	83.9792±3.0633	90.0896±2.1342	59.2292±2.8558	91.5417±1.1885
LS	67.7941±4.6967	71.0474±1.3878	80.1458±2.6877	74.3433±6.4646	29.6042±2.9236	88.0583±0.9715
ReliefF	68.6765±5.9420	70.4743±1.4343	81.8125±3.2055	86.8358±3.1096	57.1979±1.6016	91.5833±0.6052
mRMR	65.2206±6.0025	70.2273±1.8071	86.5000±2.1306	89.8507±1.7673	59.5313±2.2232	83.7875±1.3579
SBMLR	68.0147±5.1801	71.0277±1.6112	**88.3750±2.9891**	97.0090±1.2785	**60.9604±3.5565**	93.5542±0.7282
DLSR-FS	**70.0000±5.6130**	**71.4032±1.3189**	88.0000±2.0617	**97.1552±1.3230**	59.0417±2.1088	**93.8625±0.5957**

TABLE VIII

GROUP II: AVERAGE ACCURACY (%) AND THE STANDARD DEVIATION OF THE SELECTED 80 TOP FEATURES

	Coil20	Cora-OS	WebKB-CL	WebKB-WT	WebKB-WC	CAR	Glioma
MI	96.2000±0.6904	70.7161±1.9287	90.2927±2.1540	90.8664±1.0390	89.0733±1.8544	88.7963±4.2474	67.1667±7.2769
Gini	95.0756±0.8801	70.3290±1.8644	90.0976±2.1357	90.7230±1.0976	89.6067±1.7360	81.1574±5.8350	63.1667±9.9399
T-test	89.4593±1.0278	38.7677±0.2234	52.4797±28.073	24.2053±3.0866	21.8642±4.1544	57.5926±3.3127	56.6667±11.239
FS	95.7733±0.8293	70.6258±1.8905	90.0163±2.2008	89.8234±1.3854	88.7500±1.7214	84.1667±4.9627	**66.1667±9.6291**
LS	87.6337±0.9212	7.4839±0.0000	13.1707±0.0000	10.5740±18.681	9.3750±0.0000	71.9907±7.3023	62.6667±7.6929
ReliefF	93.3488±1.5234	70.0710±1.2422	90.2439±2.6535	90.0386±1.2911	89.2026±1.3948	90.0926±3.4265	62.8333±9.1303
mRMR	95.4477±0.8032	70.4903±2.0162	90.0488±1.9545	90.6788±1.1885	89.8653±1.6798	83.0093±4.5686	64.5000±8.6704
SBMLR	92.9244±4.3400	70.1871±1.7565	85.6829±6.6610	91.6225±1.8058	86.9558±3.2378	92.6389±2.4859	64.5000±9.5068
DLSR-FS	**97.3547±0.6046**	**70.9742±1.7580**	**90.8537±2.9225**	**92.1854±2.2138**	**90.2909±1.5744**	**93.9815±2.0704**	63.8333±8.0405

randomly selected as training samples, and the remaining data points are treated as the test samples. The number of training samples is listed in Table VI. The selected features and the trained classifier with the selected λ and C are used to classify the test samples.

The above training method is also used in LS to train the group of parameters σ and C. Except LS and our algorithm, for the remaining methods, only parameter C in SVM needs to be selected via cross validation. In this process, the number of features to be selected d is fixed.

C. Classification Accuracy Comparisons

Figs. 2 and 3 show the classification accuracy curves of all nine feature selection methods on the twenty data sets. The final classification accuracy is calculated as the average of the 20 trials. In total, eight different numbers of selected features are evaluated. For the two low-dimensional data sets, Protein and Vehicle, the number of features to be selected is $[2, 4, 6, \ldots, 16]$, respectively. For the remaining eighteen high-dimensional, the number of selected features is taken as $[10, 20, 30, \ldots, 80]$, respectively. For clarity, for each data set we use two figures to illustrate the accuracy and the standard deviation. Among them, the first figure shows the curves obtained with MI, Gini, T-test, FS, and our method, while the second figure shows the curves obtained with LS, ReliefF, mRMR, SBMLR, and our method. The results obtained by our method are illustrated twice for comparison.

Compared with MI, Gini, T-test, FS, LS, and ReliefF, from Figs. 2 and 3 we see that our algorithm achieves higher mean accuracies on almost all data sets with different numbers of selected features. There are only a few cases where our algorithm generates slightly low accuracy on the low-dimensional data sets. For example, on the Protein data set with $d = 12$, MI, Gini, FS, and T-Test achieve slightly higher accuracy than our algorithm. Additionally, only on one high-dimensional data set Glioma, the accuracy obtained by MI is higher than that obtained with our algorithm. Finally, it should be pointed out that the performance of LS could be largely determined on the graph construction that is related to the number of nearest neighbors as well as the edge affinities between neighboring data points (for example, on some document data sets, using the Gaussian weighting function may generate unsatisfactory results). However, how to construct a proper graph is still an open problem as the optimal one could be well tuned to explore the relations between data points.

Compared with mRMR, our algorithm achieves higher mean accuracies on most data sets with different numbers of selected features. Actually, from Figs. 2 and 3, we see that it is only on AR and CLL-SUB-111 that it clearly outperforms our algorithm. In addition, the performance of our method is comparable to that of SBMLR, which is a typical sparse feature selection approach. In contrast, on most data sets, our algorithm achieves higher mean classification accuracies.

In summary, on most data sets, the accuracy curves obtained by our algorithm locate at the top level, compared with those obtained by the remaining eight algorithms. For more clarity,

TABLE IX

GROUP III: THE AVERAGED ACCURACY (%) AND THE STANDARD DEVIATION OF THE SELECTED 80 TOP FEATURES

	Lung	MLL	SRBCT	CLL-SUB-111	GLA-BAR-180	TOX-171	TDT2
MI	89.2481±3.1249	91.5476±2.8357	79.6078±7.1805	52.2143±4.9155	55.1389±6.3049	70.6796±5.0458	**97.4306±0.9317**
Gini	86.0150±4.9117	96.1905±3.0314	**96.5686±4.0172**	54.2143±8.1405	52.1296±5.3747	67.1359±3.8997	97.2556±0.7187
T-test	87.4812±6.4415	62.9762±12.226	92.3529±3.3602	51.7143±5.5541	52.6389±5.3256	63.9320±7.4359	71.9028±0.6640
FS	86.1654±5.2682	95.5952±2.2221	96.5686±3.0426	52.7857±6.4302	51.6667±5.6009	66.8447±5.7505	96.9861±0.7501
LS	88.6842±2.6997	92.6190±3.4460	88.0392±7.7366	52.8571±4.9051	53.9815±3.9524	69.7573±4.9130	25.1806±0.2745
ReliefF	90.6391±1.5520	95.7714±2.2221	96.2745±3.9677	53.5000±8.2714	51.0185±4.4141	68.8350±5.0586	95.7222±1.2105
mRMR	89.9248±2.8594	93.0952±5.0003	89.1176±6.6865	**61.0000±5.8094**	52.4074±4.8848	73.9806±4.8961	97.2556±0.8041
SBMLR	91.6917±1.3017	93.8095±4.3971	91.2745±5.5632	50.4286±6.3567	53.4259±4.8635	63.3981±5.4111	95.9306±1.0788
DLSR-FS	**92.7820±1.5696**	**96.7857±2.0836**	96.4706±4.3427	55.9286±6.8946	**58.2870±4.9770**	**75.2427±5.2567**	97.2917±1.0310

TABLE X

RESULTS OF PAIRED STUDENTS' t TESTS ON THE ">" RELATIONSHIP BETWEEN THE TWO ACCURACIES (MEANS) REPORTED IN TABLES VII–IX

	Our > MI	Our > Gini	Our > T-test	Our > FS	Our > LS	Our > ReliefF	Our > mRMRr	Our > SBMLR
Protein	1	0	0	0	1	0	1	0
Vehicle	1	1	1	1	1	1	1	0
AT&T	1	0	1	1	1	1	1	0
Umist	1	1	1	1	1	1	1	0
AR	1	1	1	0	1	1	0	0
Usps	1	1	1	1	1	1	1	1
Coil20	1	1	1	1	1	1	1	1
Cora-OS	0	1	1	0	1	1	0	1
WebKB-CL	0	0	1	0	1	0	1	1
WebKB-WT	1	1	1	1	1	1	1	0
WebKB-WC	1	1	1	1	1	1	1	1
CAR	1	1	1	1	1	1	1	1
Glioma	0	0	1	0	0	0	0	0
Lung	1	1	1	1	1	1	1	1
MLL	1	0	1	1	1	1	1	1
SRBCT	1	0	1	0	1	0	1	1
CLL-SUB-111	1	0	1	1	1	0	0	1
GLA-BAR-180	1	1	1	1	1	1	1	1
TOX-171	1	1	1	1	1	1	0	1
TDT2	0	0	1	1	1	1	0	1

Tables VII–IX list the averaged accuracy and the standard deviation evaluated on 20 random splits. These values are obtained with $d = 16$ for Protein and Vehicle data sets, and $d = 80$ for the remaining 18 data sets. The classification accuracies in Tables VII–IX, indicate that our algorithm achieves the highest mean accuracy on 14 data sets. For the results reported in Tables VII–IX, we also performed the paired student's t test on the 20 data sets. Here, the hypothesis is "the classification (mean) accuracy obtained by DLSR-FC is greater than that obtained by the other (given) method." The results of the statistical tests are reported in X. We see the decision that "our algorithm achieves higher classification accuracy" is true with probability 0.95 on most data sets, compared with the remaining eight algorithms.

D. Computational Complexity of DLSR-FS

This section analyzes the computational complexity of DLSR-FS described in Algorithm 2. The most computationally demanding step of Algorithm 2 is step 9. It is easy to justify that calculating $\mathbf{T}_1 = \tilde{\mathbf{X}}^T\mathbf{D}\tilde{\mathbf{X}} + \lambda\Sigma$ will scale in about $O(n(m+1)^2)$, and calculating $\mathbf{T}_2 = \tilde{\mathbf{X}}^T\mathbf{D}\mathbf{T}$ will scale in about $O((m+1)nc)$. Note that computing $\mathbf{T}_3 = \mathbf{T}_1^{-1}\mathbf{T}_2$ will scale in about $o(c(m+1)^2)$ by solving linear equations. Thus, in total, the computational complexity of step 9 will be up to about $O((n+c)(m+1)^2 + (m+1)nc)$. Another computationally demanding operation is performed in step 20. The computational complexity of this step is about $O(nmc)$. All the remaining steps do not contain complex computations.

In our experiments, we must deal with high-dimensional data. For example, the dimensionality of the data points in GLA-BAR-180 is up to 49 151. It requires a lot of memory and a large amount of time to solve $\mathbf{T}_3 = \mathbf{T}_1^{-1}\mathbf{T}_2$. Actually, we can reformulate (36) to reduce the cost in the cases of $n < m$. With simple matrix operation, we have

$$(\tilde{\mathbf{X}}^T\mathbf{D}\tilde{\mathbf{X}} + \lambda\Sigma)^{-1}\tilde{\mathbf{X}}^T\mathbf{D} = \mathbf{T}_4(\tilde{\mathbf{X}}\mathbf{T}_4 + \lambda\mathbf{I}_n)^{-1} \tag{48}$$

where $\mathbf{T}_4 = \Sigma^{-1}\tilde{\mathbf{X}}^T\mathbf{D}$.

Note that in (48) both Σ and $\mathbf{D}$ are two diagonal matrices. Thus $\mathbf{T}_4$ can be easily computed. Now it is easy to check that

TABLE XI

AVERAGE TRAINING TIME (SECONDS). THE NUMBERS IN THE FIRST COLUMN CORRESPOND TO THE DATA SETS ORDERLY IN TABLE VI

	MI	Gini	Ttest	FS	LS	ReliefF	mRmr	Sbmlr	Our
1	0.001	0.05	0.002	0.01	0.04	0.02	0.97	0.05	0.05
2	0.001	0.02	0.001	0.01	0.03	0.12	0.16	0.43	0.41
3	0.19	20.9	0.01	0.82	0.05	0.69	296.	33.5	0.16
4	0.37	57.4	0.03	1.68	0.37	5.53	536.	37.1	0.97
5	0.70	151	0.01	2.89	0.26	4.97	581.	136.1	1.16
6	0.04	16.6	0.01	0.10	0.81	2.18	98.8	6.31	5.34
7	0.05	9.61	0.01	0.18	0.32	1.94	102	7.39	1.52
8	0.50	1.23	1.91	1.18	4.01	19.4	504	59.8	8.21
9	0.27	0.66	1.91	1.02	0.51	3.58	537	68.7	1.17
10	0.30	0.88	6.43	236	0.73	4.79	498	1.74	1.73
11	0.27	0.78	12.0	177	0.85	5.35	500	47.4	1.97
12	0.63	36.1	0.21	3.31	0.29	1.73	107	34.5	0.45
13	0.14	2.76	0.04	0.63	0.06	0.07	62.3	0.02	0.08
14	0.13	7.17	0.03	0.58	0.09	0.43	128	4.60	0.20
15	0.20	3.78	0.07	0.66	0.08	0.21	134	5.12	0.12
16	0.05	1.44	0.02	0.33	0.05	0.10	8.42	0.81	0.07
17	0.44	2.00	0.10	1.27	0.22	0.66	6.26	3.87	0.38
18	5.34	23.4	0.45	7.04	1.67	10.9	33.1	60.8	5.74
19	0.17	3.77	0.05	0.83	0.20	0.72	7.69	7.57	0.61
20	3.28	4.10	14.3	5.58	5.54	39.7	14.3	66.3	11.7

the computational complexity of computing $\tilde{\mathbf{W}}$ in (36) via (48) will be about $O((m+c+1)n^2+(m+1)nc)$. We see that it will be largely reduced in the case of $n \ll m$.

Table XI reports the averaged training time of running one split with the nine algorithms. The result does not include the time to train the SVM classifier. Compared with the three baseline algorithms, MI, Gini, and T-Test, our algorithm will take more time as it is implemented iteratively. Compared with the four classic algorithms, FS, ReliefF, mRMR, and SBMLR, our algorithm costs less time on most data sets, in particular on high-dimensional data sets, as for instance GLA-BAR-111 which has 49 151 dimensions. Here, our algorithm takes 5.74 s for training whereas FS, ReliefF, mRMR, and SBMLR will take about 7.04 s, 10.9 s, 33.1 s, and 60.8 s.

VI. CONCLUSION

In this paper, we presented a framework of DLSR for multiclass classification and feature selection. The main novel contributions of this paper were the following: 1) a training model with compact form for DLSR was developed, which translated the one-versus-rest training rule for multiclass classification problems well; 2) based on this compact model, a learning framework with sparse representation both on the LSR term and on the regularization term was constructed for feature selection; and 3) the homogeneous coordinate representation for the LSR yielded an effective and efficient solution to the feature selection formulation with sparse representation. Additionally, theoretical analyses about the derived model for multiclass classification and its extension for feature selection were given. Comparative experiments indicated that the proposed method results in high classification accuracy, convergence to global optimum, and effectiveness for high-dimensional data.

The main limit of our approach is that the training time is still high compared with the traditional feature selection approaches. The optimization problem is actually divided into two subproblems, which are solved one by one in an iterative optimization framework. Although the subproblems are all convex and the convergency of the algorithm can be guaranteed, the convergence speed is slow. In the future, we would like to speed up our algorithm by combining it with a multiple gradient descent technique.

ACKNOWLEDGMENT

The authors would like to thank the associate editor and the anonymous reviewers, who helped to greatly improve the quality and presentation of this paper.

REFERENCES

[1] T. Strutz, *Data Fitting and Uncertainty: A Practical Introduction to Weighted Least Squares and Beyond*. Wiesbaden, Germany: Vieweg, 2010.

[2] S. Wold, H. Ruhe, H. Wold, and W. Dunn, III, "The collinearity problem in linear regression, the partial least squares (PLS) approach to generalized inverse," *J. Sci. Stat. Comput.*, vol. 5, no. 3, pp. 735–743, 1984.

[3] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[4] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press, 2006, pp. 1473–1480.

[5] P. Schneider, K. Bunte, H. Stiekema, B. Hammer, T. Villmann, and M. Biehl, "Regularization in matrix relevance learning," *IEEE Trans. Neural Netw.*, vol. 21, no. 5, pp. 831–840, May 2010.

[6] J. Leski, "Ho–Kashyap classifier with generalization control," *Pattern Recognit. Lett.*, vol. 24, no. 14, pp. 2281–2290, 2003.

[7] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[8] H. Liu and H. Motoda, *Computational Methods of Feature Selection*. New York: Chapman & Hall, 2007.

[9] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2006, pp. 507–514.

[10] Z. Zhao and H. Liu, "Semi-supervised feature selection via spectral analysis," in *Proc. SIAM Conf. Data Mining*, Minneapolis, MN, 2007, pp. 641–646.

[11] Z. Xu, I. King, M. R.-T. Lyu, and R. Jin, "Discriminative semi-supervised feature selection via manifold regularization," *IEEE Trans. Neural Netw.*, vol. 21, no. 7, pp. 1033–1047, Jul. 2010.

[12] M. Kalakech, P. Biela, L. Macaire, and D. Hamad, "Constraint scores for semi-supervised feature selection: A comparative study," *Pattern Recognit. Lett.*, vol. 32, no. 5, pp. 656–665, 2011.

[13] L. Zhou, L. Wang, and C. Shen, "Feature selection with redundancy-constrained class separability," *IEEE Trans. Neural Netw.*, vol. 21, no. 5, pp. 853–858, May 2010.

[14] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[15] P. A. Estevez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Trans. Neural Netw.*, vol. 20, no. 2, pp. 189–201, Feb. 2009.

[16] F. Oveisi, S. Oveisi, A. Efranian, and I. Patras, "Tree-structured feature extraction using mutual information," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 127–137, Jan. 2012.

[17] J. Li, M. T. Manry, P. L. Narasimha, and C. Yu, "Feature selection using a piecewise linear network," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1101–1115, Sep. 2006.

[18] E. Romero and J. M. Sopena, "Performing feature selection with multilayer perceptrons," *IEEE Trans. Neural Netw.*, vol. 19, no. 3, pp. 431–441, Mar. 2008.

[19] C. Bhattacharyya, "Second order cone programming formulations for feature selection," *J. Mach. Learn. Res.*, vol. 5, pp. 1417–1433, Dec. 2004.

[20] J.-B. Yang and C.-J. Ong, "Feature selection using probabilistic prediction of support vector regression," *IEEE Trans. Neural Netw.*, vol. 22, no. 6, pp. 954–962, Jun. 2011.

[21] G. Fung and O. L. Mangasarian, "Data selection for support vector machine classifiers," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, Boston, MA, 2000, pp. 64–70.

[22] A. Y. Ng, "Feature selection, ℓ_1 versus ℓ_2 regularization, and rotational invariance," in *Proc. Int. Conf. Mach. Learn.*, Banff, AB, Canada, 2004, pp. 78–85.

[23] G. C. Cawley, N. L. C. Talbot, and M. Girolami, "Sparse multinomial logistic regression via Bayesian ℓ_1 regularisation," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2006, pp. 209–216.

[24] L. Wang, J. Zhu, and H. Zou, "Hybrid huberized support vector machines for microarray classification," in *Proc. Int. Conf. Mach. Learn.*, Corvalis, OR, 2007, pp. 983–990.

[25] F. Nie, H. Huang, X. Cai, and C. Ding, "Efficient and robust feature selection via joint ℓ_2, 1-norms minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2010, pp. 1813–1821.

[26] M. Tan, L. Wang, and I. W. Tsang, "Learning sparse SVM for feature selection on very high dimensional datasets," in *Proc. Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 1047–1054.

[27] T. Jebara, "Multitask sparsity via maximum entropy discrimination," *J. Mach. Learn. Res.*, vol. 12, pp. 75–110, Jan. 2011.

[28] L. W. Zhong and J. T. Kwok, "Efficient sparse modeling with automatic feature grouping," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 9, pp. 1436–1447, Sep. 2012.

[29] P. Somol and J. Novovicova, "Evaluating stability and comparing output of feature selectors that optimize feature subset cardinality," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 11, pp. 1921–1939, Nov. 2010.

[30] Y. Jiang and J. Ren, "Eigenvalue sensitive feature selection," in *Proc. Int. Conf. Mach. Learn.*, Bellevue, WA, 2011, pp. 89–96.

[31] C. M. Bishop, *Neural Networks for Pattern Recognition*. Cambridge, U.K.: Oxford Univ. Press, 1995.

[32] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for SVMs," in *Proc. Adv. Neural Inf. Process. Syst.*, Denver, CO, 2000, pp. 668–674.

[33] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, nos. 1–3, pp. 389–422, 2002.

[34] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-norm support vector machines," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2004, pp. 1–8.

[35] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal Stat. Soc., Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.

[36] P. S. Bradley and O. L. Mangasarian, "Feature selection via concave minimization and support vector machines," in *Proc. Int. Conf. Mach. Learn.*, Madison, WI, 1998, pp. 82–90.

[37] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*, 2nd ed. New York: Wiley, 2000.

[38] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[39] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer-Verlag, 2000.

[40] L. Chen, I. W. Tsang, and D. Xu, "Laplacian embedded regression for scalable manifold regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 902–915, Jun. 2012.

[41] S. Xiang, F. Nie, C. Pan, and C. Zhang, "Regression reformulations of LLE and LTSA with locally linear transformation," *IEEE Trans. Syst., Man Cybern., B, Cybern.*, vol. 42, no. 5, pp. 1250–1262, Oct. 2011.

[42] T. Cover and J. M. van Campenhout, "On the possible orderings in the measurement selection problem," *IEEE Trans. Syst., Man Cybern.*, vol. 7, no. 9, pp. 657–661, Sep. 1977.

[43] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[44] E. J. Candes and J. Romberg, "Quantitative robust uncertainty principles and optimally sparse decompositions," *Found. Comput. Math.*, vol. 6, no. 2, pp. 227–254, 2006.

[45] C. H. Q. Ding, D. Zhou, X. He, and H. Zha, "R1-PCA: Rotational invariant ℓ_1-norm principal component analysis for robust subspace factorization," in *Proc. Int. Conf. Mach. Learn.*, Pittsburgh, PA, 2006, pp. 281–288.

[46] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. 2nd IEEE Workshop Appl. Comput. Vis.*, Dec. 1994, pp. 138–142.

[47] A. Martinez and R. Benavente, "The AR face database," Computer Vision Center, Univ. Autònoma de Barcelona, Bellaterra, Barcelona, Tech. Rep. 24, Jun. 1998.

[48] S. Nene, S. Nayar, and H. Murase, "Columbia object image library (COIL-20)," Dept. Comput. Sci., Columbia Univ., New York, Tech. Rep. CUCS-006-96, 1996.

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

[50] C.-C. Chang and C.-J. Lin. (2001). *LIBSVM: A Library for Support Vector Machines* [Online]. Available: http://www.csie.ntu.edu.tw/∼cjlin/limbos

[51] C. L. Nutt, D. R. Mani, R. A. Betensky, P. Tamayo, J. G. Cairncross, C. Ladd, U. Pohl, C. Hartmann, M. E. McLaughlin, T. T. Batchelor, P. M. Black, A. von Deimling, S. L. Pomeroy, T. R. Golub, and D. N. Louis, "Gene expression-based classification of malignant gliomas correlates better with survival than histological classification," *Cancer Res.*, vol. 63, pp. 1602–1607, Apr. 2003.

[52] A. Bhattacharjee, W. G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E. J. Mark, E. S. Lander, W. Wong, B. E. Johnson, T. R. Golub, D. J. Sugarbaker, and M. Meyerson, "Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses," *Proc. Nat. Acad. Sci.*, vol. 98, no. 24, pp. 13790–13795, 2001.

[53] Z. Cai, R. Goebel, M. Salavatipour, Y. Shi, L. Xu, and G.-H. Lin, "Selecting genes with dissimilar discrimination strength for sample class prediction," in *Proc. 5th Asia-Pacific Bioinf. Conf.*, Hong Kong, 2007, pp. 81–90.

[54] J. Khan, J. S. Wei, M. Ringner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, and P. S. Meltzer, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature Med.*, vol. 7, no. 6, pp. 673–679, 2001.

[55] C. Haslinger, N. Schweifer, S. Stilgenbauer, H. Dohner, and P. Lichter, "Microarray gene expression profiling of B-cell chronic lymphocytic leukemia subgroups defined by genomic aberrations and VH mutation status," *J. Clin. Oncol.*, vol. 22, no. 19, pp. 3937–3949, 2004.

[56] L. Sun, A.-M. Hui, Q. Su, A. Vortmeyer, Y. Kotliarov, S. Pastorino, A. Passaniti, J. Menon, J. Walling, R. Bailey, M. Rosenblum, T. Mikkelsen, and H. A. Fine, "Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain," *Cancer Cell*, vol. 9, no. 4, pp. 287–300, 2006.

[57] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[58] L. Cehovin and Z. Bosnic, "Empirical evaluation of feature selection methods in classification," *Intell. Data Anal.*, vol. 14, no. 3, pp. 265–281, Aug. 2010.

[59] D. C. Montgomery and D. C. Hubele, *Engineering Statistics*. Hoboken, NJ: Wiley, 2007.

[60] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proc. 9th Int. Workshop Mach. Learn.*, Aberdeen, U.K., 1992, pp. 249–256.

Shiming Xiang received the B.S. degree in mathematics from Chongqing Normal University, Chongqing, China, the M.S. degree from Chongqing University, Chongqing, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 1993, 1996, and 2004, respectively.

He was a Lecturer with the Huazhong University of Science and Technology, Wuhan, China, from 1996 to 2001. He was a Post-Doctoral Fellow with the Department of Automation, Tsinghua University, Beijing, until 2006. He is currently an Associate Professor with the Institute of Automation, Chinese Academy of Sciences. His current interests include image processing, pattern recognition, and machine learning.

Feiping Nie received the B.S. degree in computer science from the North China University of Water Conservancy and Electric Power, Beijing, China, the M.S. degree from Lanzhou University, Lanzhou, China, and the Ph.D. degree from the Department of Automation, Tsinghua University, Beijing, in 2000, 2003, and 2009, respectively.

He is currently a Research Assistant Professor with the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington. His current research interests include machine learning and its applications.

Gaofeng Meng received the B.S. degree in applied mathematics from Northwestern Polytechnical University, Xi'an, China, and the M.S. degree in applied mathematics from Tianjin University, Tianjin, China, and the Ph.D. degree in control science and engineering from Xi'an Jiaotong University, Xi'an, in 2002, 2005, and 2009, respectively.

He has been with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, since 2009, where he is currently an Assistant Professor. His current research interests include image processing, computer vision, and pattern recognition.

Chunhong Pan received the B.S. degree in automatic control from Tsinghua University, Beijing, China, the M.S. degree from the Shanghai Institute of Optics and Fine Mechanics, Chinese Academy of Sciences, Beijing, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, in 1987, 1990, and 2000, respectively.

He is currently a Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His current research interests include computer vision, image processing, computer graphics, and remote sensing.

Changshui Zhang (M'02) received the B.S. degree in mathematics from Peking University, Beijing, China, and the Ph.D. degree from Tsinghua University, Beijing, in 1986 and 1992, respectively.

He joined the Department of Automation, Tsinghua University, in 1992, where he is currently a Professor. He has authored over 200 papers published in journals and conferences. His current research interests include pattern recognition and machine learning.

Dr. Zhang is currently on the editorial board of *Pattern Recognition*.