Efficient Image Classification via Multiple Rank Regression

Chenping Hou, *Member, IEEE*, Feiping Nie, Dongyun Yi, and Yi Wu

***Abstract*— The problem of image classification has aroused considerable research interest in the field of image processing. Traditional methods often convert an image to a vector and then use a vector-based classifier. In this paper, a novel multiple rank regression model (MRR) for matrix data classification is proposed. Unlike traditional vector-based methods, we employ multiple-rank left projecting vectors and right projecting vectors to regress each matrix data set to its label for each category. The convergence behavior, initialization, computational complexity, and parameter determination are also analyzed. Compared with vector-based regression methods, MRR achieves higher accuracy and has lower computational complexity. Compared with traditional supervised tensor-based methods, MRR performs better for matrix data classification. Promising experimental results on face, object, and hand-written digit image classification tasks are provided to show the effectiveness of our method.**

***Index Terms*— Dimensionality reduction, image classification, multiple rank regression, tensor analysis.**

I. Introduction

IMAGE DATA, or more commonly, tensor data, have been emerged in various applications. For example, in biologic data mining, the data, such as face images [1], palm images [2], or MRI data [3], are usually represented in the form of data matrices. Additionally, in video data mining, the data in each time frame is also a matrix. How to classify this kind of data is one of the most important topics for both image processing and machine learning.

In the literature, a lot of classification approaches have been proposed, such as K-Nearest Neighborhoods classifier (KNN) [4], Support Vector Machine (SVM) [5] and one-dimensional Regression methods (for brief, denoted as 1DREG in the following) [6]. Some of them are similarity based, such as KNN. Some of them are margin based, such as SVM. Among these approaches, due to their simplicity, effectiveness, and inductive nature, regression methods have been widely used in many real applications [6]. Nevertheless, traditional classification methods are usually vector-based. They assume that the inputs of an algorithm are all vectors. When they are employed to manipulate matrix data, we have to transform matrix data into vector data at first. One common way is reformulating a vector by connecting each row (or column) of a matrix. For example, as shown in the top plane of Fig. 1, an image of size $m \times n$ has be reformulated as a mn-dimensional vector and then the traditional regression method can be used.

Although the performances of traditional classification approaches are prominent in many cases, they may be lack of efficiency in managing matrix data. The reasons mainly include: (1) When we reformulate an image matrix as a vector, the dimensionality of this vector is often very high. For example, for a small image of resolution 100×100, the reformulated vector is 10 000 dimensional. The performances of these methods will degrade due to the increase of dimensionality [7]. (2) With the increase of dimensionality, the computational time will increase drastically. If the matrix scale is a little larger, traditional approaches can not be implemented in this scenario [8]. (3) When a matrix is expanded as a vector, we would lose the correlations of the matrix data [9]. For example, the $m \times n$ matrix representation of an image suggests that the real number of freedom is far less than mn, which is revealed by representing an image as a mn-dimensional vector. This will lead to the over fitting problem in regression [6].

To solve these problems, there have been many feature based methods, such as the elastic graph model which can remain spatial information in a compact dimensionality [10]–[12]. Recently, a lot of interests have been conducted on tensor-based approaches for matrix data analysis. Intrinsically, a matrix is a two-order tensor. Vasilescu and Terzopoulos have firstly proposed a novel tensor face for face recognition [13]. Other researchers have also extended a lot of traditional subspace learning methods, such as Principal Component Analysis (PCA) [14], Linear Discriminant Analysis (LDA) [15], Locality Preserving Projections (LPP) [16], [17], etc, into their tensor counterparts [18]–[23]. Nevertheless, the purpose in designing these approaches is to learn subspaces of original matrix data. For matrix data classification, one possible way in using these methods is projecting matrix data into a subspace at first and then employing another classifier. Besides, among these approaches, Two-dimensional Linear Discriminant Analysis (2DLDA) [19] is a popular supervised tensor based approach. It uses label information

Manuscript received September 25, 2011; revised June 30, 2012; accepted August 6, 2012. Date of publication August 17, 2012; date of current version December 20, 2012. This work was supported by the National Basic Research Program of China (973 Program) under Grant 2009CB320602 and the National Science Foundation of China under Grant 61005003 and Grant 60975038. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Kenneth Lam.

C. Hou, D. Yi, and Y. Wu are with the Department of Mathematics and Systems Science, National University of Defense Technology, Changsha 410073, China (e-mail: hcpnudt@gmail.com; dongyun.yi@gmail.com; wuyi_work@sina.com).

F. Nie is with the Department of Computer Science and Engineering, University of Texas, Arlington, TX 76010 USA (e-mail: feipingnie@gmail.com).

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/TIP.2012.2214044

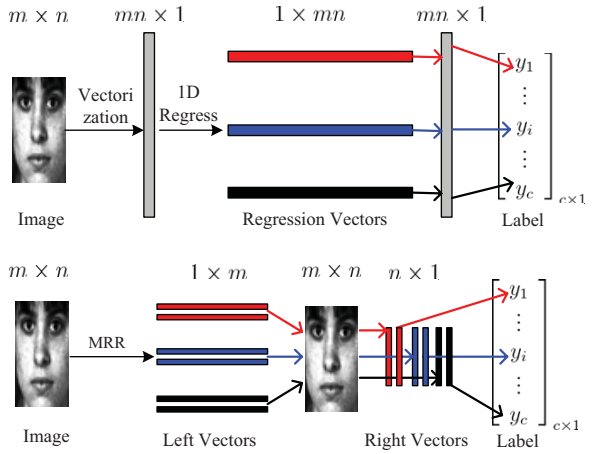

Fig. 1. Intuition of multiple rank regression. The top procedure is traditional regression and the bottom is multiple rank regression.

in manipulating matrix data. Moreover, Gabriel et al have analyzed the Generalized Bilinear Regression (GBR) model [24] mathematically. They have not used it in classifying matrix data. When these tensor-based approaches are employed to classify matrix data, their performances can also be improved since (1) Most of them are unsupervised, they can not be used for tensor data classification directly. (2) Although 2DLDA is supervised, it uses label information to learn the subspace. It can not be used for classification directly. (3) Traditional regression method, such as GBR, is analyzed mathematically, it has not been used for matrix data classification. Besides, GBR only uses one left projecting vector together with one right projecting vector. Its fitting error is too large for real regression problem. We will also show that it is a special case of our algorithm essentially. (See Section IV (D).)

In this paper, we introduce a novel matrix classification model using multiple rank regression. Instead of expanding an $m \times n$ matrix to an mn-dimensional vector as in traditional regression methods, we employ two groups of transformation matrices for linear regression. More importantly, there are several transformation vectors in each group. As shown in the bottom of Fig. 1, each data is regressed to its label vector by using multiple rank left and right regression vectors. In other words, we approximate each data by multiple rank regression vectors. Compared with other vector based methods, e.g. 1DREG, and tensor-based approaches, such as 2DLDA and GBR, it is more suitable for matrix data classification. Some deep analyses in Section IV (D) indicate that MRR can be regarded as a tradeoff between the capacity of learning and generalization for regression. We have also proposed an effective optimization strategy. Compared with traditional vector based classification approaches, the computational complexity of MRR is lower. Plenty of experiments on different kinds of data are presented for illustration.

The rest of this paper is organized as follows. Section II will provide some notations and some related works. We formulate the proposed MRR algorithm and provide an effective way in solving this problem in Section III. The performance analyses, including convergence behavior, initialization, computational complexity and parameter determination, are presented in Section IV. Section V provides some promising comparing results on various kinds of data sets, followed by the conclusions and future works in Section VI.

TABLE I
NOTATIONS

m	The first dimensionality of matrix data
n	The second dimensionality of matrix data
l	Number of training points
c	Number of class
k	Rank of regression
t	Number of testing points
$\mathbf{e} = [1, 1, \ldots, 1] \in \mathbb{R}^{1\times l}$	The row vector of all ones
$\mathbf{X}_i \in \mathbb{R}^{m\times n}$	The ith training matrix data
$\mathbf{x}_i \in \mathbb{R}^{mn}$	The vectorization of $\mathbf{X}_i$
$\mathbf{y}_i \in \mathbb{R}^{c}$	The label vector of $\mathbf{X}_i$
$\mathbf{u}_j^{(r)} \in \mathbb{R}^{m}$	The jth left regression vector for class r
$\mathbf{v}_j^{(r)} \in \mathbb{R}^{n}$	The jth right regression vector for class r
b_r	The constant regression value for class r
$\mathbf{U}^{(r)} = [\mathbf{u}_1^{(r)}, \ldots, \mathbf{u}_k^{(r)}]$	The left regression matrix for class r
$\mathbf{V}^{(r)} = [\mathbf{v}_1^{(r)}, \ldots, \mathbf{v}_k^{(r)}]$	The right regression matrix for class r

II. Related Work

In this section, we will introduce three representative methods, including two supervised two-dimensional methods, i.e., 2DLDA and GBR, and one-dimensional regression method, i.e., 1DREG. The reason why we choose these methods is that they have close relationship with our method. Before going into the details, let us introduce the notations at first.

A. Notation

As mentioned above, we try to solve a supervised matrix data classification problem in this paper. Let $\{\mathbf{X}_i \in \mathbb{R}^{m\times n} | i = 1, 2, \ldots, l\}$ as the set of training examples and the associated class label vectors are $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_l\} \subset \mathbb{R}^c$. Here, $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{ic}]^T$. $y_{ij} = 1$ if and only if $\mathbf{X}_i$ belongs to the j-th category and $y_{ij} = 0$ otherwise. c is the number of classes. m and n are the first and second dimensions of each image data. l is the number of training points.

Define $\mathbf{e} = [1, 1, \ldots, 1] \in \mathbb{R}^{1\times l}$ as a row vector of all ones and $\alpha > 0$ as a balance parameter. Define $\mathbf{u}_j^{(r)} \in \mathbb{R}^m$ and $\mathbf{v}_j^{(r)} \in \mathbb{R}^n$ as the j-th left and right regression vectors for class r. Other important notations are summarized in Table I. See more details there and we will explain their concrete meanings when it is firstly used.

B. 2DLDA

2DLDA is one of the representative methods for the task of supervised two dimensional subspace learning. It is a two dimensional extension of traditional LDA approach. Let $\mathcal{M}_i$ be the set of training points in the i-th class, where $\mathcal{M}_i$ has l_i data. Let $\bar{\mathbf{X}}_i = \frac{1}{l_i}\sum_{\mathbf{X}_j\in\mathcal{M}_i} \mathbf{X}_j$ be the mean of the i-th class for $1 \le i \le c$, and $\bar{\mathbf{X}} = \frac{1}{l}\sum \mathbf{X}_i$ be the global mean.

2DLDA aims to find two transformation matrices $\mathbf{L}$ and $\mathbf{R}$, which map $\mathbf{X}_i$ to its low dimensional embedding, i.e., $\mathbf{Z}_i$, by

$\mathbf{Z}_i = \mathbf{L}^T\mathbf{X}_i\mathbf{R}$. It tries to minimize the within-class distance D_w and maximize the between-class distances D_b, which are defined as follows

$$D_w = Tr\left(\sum_{j=1}^{c}\sum_{\mathbf{X}_i\in\mathcal{M}_j}\mathbf{L}^T(\mathbf{X}_i-\bar{\mathbf{X}}_j)\mathbf{R}\mathbf{R}^T(\mathbf{X}_i-\bar{\mathbf{X}}_j)^T\mathbf{L}\right),$$
$$D_b = Tr\left(\sum_{j=1}^{c} l_j\mathbf{L}^T(\bar{\mathbf{X}}_j-\bar{\mathbf{X}})\mathbf{R}\mathbf{R}^T(\bar{\mathbf{X}}_j-\bar{\mathbf{X}})^T\mathbf{L}\right). \quad (1)$$

Since it is difficult to derive the optimal $\mathbf{L}$ and $\mathbf{R}$ simultaneously, 2DLDA solves the above problem in Eq. (1) in an alternative way. Briefly, it fixes $\mathbf{L}$ in computing $\mathbf{R}$ and fixes $\mathbf{R}$ in computing $\mathbf{L}$. See more details in [19].

C. 1DREG

1DREG is a representative method in vector-based regression works. It is also a famous model for classification. Denote the matrix data $\mathbf{X}_i$ as an mn-dimensional vector data $\mathbf{x}_i$ by connecting each row (or column). 1DREG aims to regress each data to its label vector by computing c transformation vectors and constants, denoted as $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_c] \in \mathbb{R}^{mn\times c}$ and $\mathbf{b} = [b_1, b_2, \ldots, b_c]^T$. In order to avoid over fitting, we often add a regularizer. The most commonly used one is the Tikhonov regularization [25]. Briefly, the objective function of 1DREG with Tikhonov regularization is

$$\mathcal{L}(\mathbf{W},\mathbf{b}) = \textstyle\sum_{i=1}^{l}\|\mathbf{W}^T\mathbf{x}_i+\mathbf{b}-\mathbf{y}_i\|_F^2 + \alpha\|\mathbf{W}\|_F^2 \quad (2)$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix.

After some simple derivation, the optimization problem can be solved in a closed form. See more details in [25].

As seen from the formulation in Eq. (2), 1DREG converts the matrix data into a vector. Thus, it will lose the correlation of matrix data and its computational time consuming is unacceptable if the matrix scale is large.

D. GBR

As mentioned in [24], GBR is the two-dimensional counterpart of 1DREG. It replaces the regression function of 1DREG by a binlinear regression function. More concretely, in two-class scenario, we assume the left and right projection vectors are $\mathbf{u}$ and $\mathbf{v}$. Its objective function is

$$\mathcal{L}(\mathbf{u},\mathbf{v},b) = \sum_{i=1}^{l}\|\mathbf{u}^T\mathbf{X}_i\mathbf{v}+b-y_i\|^2$$

GBR has only been analyzed mathematically. Besides, it only uses one left projecting vector together with one right projecting vector. Its fitting error is too large for some real regression problems.

III. Multiple Rank Regression

We would like to describe our MRR model in this section. First, we formulate our MRR algorithm and then propose an effective method to search the approximated solution. Since the problem is not convex, we solve it in an alternative way. To show the effectiveness theoretically, we provide some deep analyses in next section.

A. Formulation

We now introduce our algorithm formally. Considering that regression methods perform well in training classifier and tensor-based methods, such as 2DLDA, are faster, we would like to integrate their merits in formulating our algorithm.

There are mainly two objective functions of MRR. Generally, the first one measures the loss with multiple rank regression and the other is the regularization term. Before going into the details, we would like to reformulate the objective function of Tikhonov regularized least square regression in Eq. (2) in another form.

After some simple deductions, the objective function in Eq. (2) can be reformulated as follows

$$\begin{aligned}\mathcal{L}(\mathbf{W},\mathbf{b}) &= \sum_{i=1}^{l}\|\mathbf{W}^T\mathbf{x}_i+\mathbf{b}-\mathbf{y}_i\|_F^2+\alpha\|\mathbf{W}\|_F^2\\ &= \sum_{i=1}^{l}\sum_{r=1}^{c}(\mathbf{w}_r^T\mathbf{x}_i+b_r-y_{ir})^2+\alpha\sum_{r=1}^{c}(\mathbf{w}_r^T\mathbf{w}_r)\\ &= \sum_{r=1}^{c}\sum_{i=1}^{l}(\mathbf{w}_r^T\mathbf{x}_i+b_r-y_{ir})^2+\alpha\sum_{r=1}^{c}(\mathbf{w}_r^T\mathbf{w}_r)\\ &= \sum_{r=1}^{c}\left(\sum_{i=1}^{l}(\mathbf{w}_r^T\mathbf{x}_i+b_r-y_{ir})^2+\alpha(\mathbf{w}_r^T\mathbf{w}_r)\right). \quad (3)\end{aligned}$$

As seen from Eq. (3), it is clear that this regression model is a combination of multiple two-category classifiers via *one versus rest* strategy. More concretely, in training the classifier for the r-th category, the labels for the points who belong to the r-th category are one. If a point does not belong to this class, its label is zero. Moreover, this training process is separate. We can regard it as c independent procedures, in which we only compute the corresponding $\mathbf{w}_r$ and b_r for $r = 1, 2, \ldots, c$. In other words, the formulation in Eq. (3) can be regarded as training c classifiers for c categories separately.

Evoked by the above formulation, we only consider to train a classifier for the r-th category. One direct way in constructing the loss function is to replace the traditional projection term, i.e., $\mathbf{w}_r^T\mathbf{x}_i$ in Eq. (3), by its tensor counterpart, such as $\mathbf{u}_r^T\mathbf{X}_i\mathbf{v}_r$, where $\mathbf{u}_r$ and $\mathbf{v}_r$ are the left and right transformation vectors for the r-th category. Nevertheless, as mentioned above, this kind of replacement will induce some constraints [9]. For example, the mn values in a tensor base $\mathbf{u}_r\mathbf{v}_r^T$ only have $m+n$ degrees of freedom. In many real cases, these added constraints are too strict. It can not characterize the original data and thus, increase the regression error.

To solve this problem, instead of using merely one couple of projecting vectors, i.e., the left projecting vector $\mathbf{u}_r$ and right projecting vector $\mathbf{v}_r$ for the r-th classifier, we propose to use k couples of left projecting vectors and right projecting vectors. They are denoted as $\{\mathbf{u}_j^{(r)}\}_{j=1}^k$ and $\{\mathbf{v}_j^{(r)}\}_{j=1}^k$. The intuition is shown in the bottom of Fig. 1. Compared with the employment of only one couple of projecting vectors, there are several advantages of our method. (1) Since we have multiple rank projecting vectors, the above mentioned constraints will be relaxed to some extent and consequently, the joint effects of these projections will decrease the regression error since one

couple of projecting vectors is a special case of our setting when $\mathbf{u}_j = \mathbf{0}$, $\mathbf{v}_j = \mathbf{0}$ for $j \geq 2$. (2) As what we will explain later, k is the parameter to balance the capacity of learning and generalization. GBR is the special case of MRR when $k = 1$. 1DREG also has close relationship with MRR when $k = \min(m, n)$.

Formally, the first loss function to train the r-th classifier is

$$\sum_{i=1}^{l}\left(\sum_{j=1}^{k}(\mathbf{u}_j^{(r)})^T\mathbf{X}_i\mathbf{v}_j^{(r)} + b_r - y_{ir}\right)^2. \tag{4}$$

where b_r is the unknown constant for the r-th category.

Denote $\mathbf{U}^{(r)} = [\mathbf{u}_1^{(r)}, \mathbf{u}_2^{(r)}, \ldots, \mathbf{u}_k^{(r)}] \in \mathbb{R}^{m\times k}$ and $\mathbf{V}^{(r)} = [\mathbf{v}_1^{(r)}, \mathbf{v}_2^{(r)}, \ldots, \mathbf{v}_k^{(r)}] \in \mathbb{R}^{n\times k}$, the loss function in Eq. (4) can be reformulated as follows

$$\sum_{i=1}^{l}\left(Tr\left(\left(\mathbf{U}^{(r)}\right)^T\mathbf{X}_i\mathbf{V}^{(r)}\right) + b_r - y_{ir}\right)^2. \tag{5}$$

The second objective function is regularization term. Considering the form of regularization for the r-th category in Eq. (3), we have

$$\|\mathbf{w}_r\|_F^2 = \mathbf{w}_r^T\mathbf{w}_r = Tr\left(\mathbf{w}_r\mathbf{w}_r^T\right). \tag{6}$$

Notice that, for the r-th classifier,

$$\begin{aligned}
&\sum_{j=1}^{k}\left(\mathbf{u}_j^{(r)}\right)^T\mathbf{X}_i\mathbf{v}_j^{(r)} = \sum_{j=1}^{k}Tr\left(\left(\mathbf{u}_j^{(r)}\right)^T\mathbf{X}_i\mathbf{v}_j^{(r)}\right)\\
&= \sum_{j=1}^{k}Tr\left(\mathbf{X}_i\mathbf{v}_j^{(r)}(\mathbf{u}_j^{(r)})^T\right) = Tr\left(\mathbf{X}_i\left(\sum_{j=1}^{k}\mathbf{v}_j^{(r)}(\mathbf{u}_j^{(r)})^T\right)\right)\\
&= Tr\left(\mathbf{X}_i\mathbf{V}^{(r)}\left(\mathbf{U}^{(r)}\right)^T\right) = Tr\left(\left(\mathbf{U}^{(r)}\left(\mathbf{V}^{(r)}\right)^T\right)^T\mathbf{X}_i\right)\\
&= \left(Vec\left(\mathbf{U}^{(r)}\left(\mathbf{V}^{(r)}\right)^T\right)\right)^T Vec(\mathbf{X}_i)
\end{aligned} \tag{7}$$

Here $Vec(\mathbf{U}^{(r)}(\mathbf{V}^{(r)})^T)$ is the vectorization of matirix $\mathbf{U}^{(r)}(\mathbf{V}^{(r)})^T$ (by connecting each row of $\mathbf{U}^{(r)}(\mathbf{V}^{(r)})^T$) and similarly for $Vec(\mathbf{X}_i)$.

Comparing the first objective function of MRR (in Eq. (4)) with 1DREG (in Eq. (3)), we know that the tensor counterpart of $\mathbf{w}_r$ in Eq. (3) is $\mathbf{U}^{(r)}(\mathbf{V}^{(r)})^T$. Notice that in 1DREG, the regularizer is $Tr(\mathbf{w}_r\mathbf{w}_r^T)$. Thus, the corresponding regularization can be regarded as $Tr(\mathbf{U}^{(r)}(\mathbf{V}^{(r)})^T\mathbf{V}^{(r)}(\mathbf{U}^{(r)})^T)$. Nevertheless, if we use this regularization directly, the formulated problem is very hard to solve. Thus, we approximate it by a simple form.

For the r-th classifier, considering the essence of Tikhonov regularization, we assume that the regularization term is defined as follows

$$\sum_{j=1}^{k}Tr\left(\mathbf{u}_j^{(r)}\left(\mathbf{v}_j^{(r)}\right)^T\mathbf{v}_j^{(r)}\left(\mathbf{u}_j^{(r)}\right)^T\right) = \sum_{j=1}^{k}\left\|\mathbf{u}_j^{(r)}\left(\mathbf{v}_j^{(r)}\right)^T\right\|_F^2. \tag{8}$$

Formally, the regularization in Eq. (8) is also similar to the corresponding Tikhonov regularizer in Eq. (6).

In summary, by combing the two objective functions in Eq. (5) and Eq. (8), we formulate our MRR algorithm as follows

$$\begin{aligned}
&\mathcal{L}\left(\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(c)}, \mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(c)}, \mathbf{b}\right)\\
&= \sum_{r=1}^{c}\sum_{i=1}^{l}\left(Tr\left(\left(\mathbf{U}^{(r)}\right)^T\mathbf{X}_i\mathbf{V}^{(r)}\right) + b_r - y_{ir}\right)^2\\
&\quad +\alpha\sum_{r=1}^{c}\sum_{j=1}^{k}Tr\left(\mathbf{u}_j^{(r)}\left(\mathbf{v}_j^{(r)}\right)^T\mathbf{v}_j^{(r)}\left(\mathbf{u}_j^{(r)}\right)^T\right)\\
&= \sum_{r=1}^{c}\left[\sum_{i=1}^{l}\left(Tr\left(\left(\mathbf{U}^{(r)}\right)^T\mathbf{X}_i\mathbf{V}^{(r)}\right) + b_r - y_{ir}\right)^2\right.\\
&\quad \left. + \alpha\sum_{j=1}^{k}Tr\left(\mathbf{u}_j^{(r)}\left(\mathbf{v}_j^{(r)}\right)^T\mathbf{v}_j^{(r)}\left(\mathbf{u}_j^{(r)}\right)^T\right)\right].
\end{aligned} \tag{9}$$

where α is also a parameter to balance the influence of two different objectives.

B. Solution

In this section, we will try to find approximated solution to the proposed problem in Eq. (9) since it is not convex. Let us simplify this problem at first.

Considering the optimization problem in Eq. (9), we know that it can be divided into c separate regression problems since the computation of $\mathbf{U}^{(r)}$ and $\mathbf{V}^{(r)}$ has no relationship with $\{\mathbf{U}^{(i)}\}_{i\neq r}$ and $\{\mathbf{V}^{(i)}\}_{i\neq r}$. In the following, without loss of generality, we only focus on deriving $\mathbf{U}^{(r)}$, $\mathbf{V}^{(r)}$ and b_r for the r-th category. The corresponding objective function is

$$\begin{aligned}
&\arg\min\mathcal{L}\left(\mathbf{U}^{(r)}, \mathbf{V}^{(r)}, b_r\right)\\
&= \sum_{i=1}^{l}\left(Tr\left(\left(\mathbf{U}^{(r)}\right)^T\mathbf{X}_i\mathbf{V}^{(r)}\right) + b_r - y_{ir}\right)^2\\
&\quad +\alpha\sum_{j=1}^{k}Tr\left(\mathbf{u}_j^{(r)}\left(\mathbf{v}_j^{(r)}\right)^T\mathbf{v}_j^{(r)}\left(\mathbf{u}_j^{(r)}\right)^T\right).
\end{aligned} \tag{10}$$

Since the problem in Eq. (10) is not jointly convex with respect to $\mathbf{U}^{(r)}$ and $\mathbf{V}^{(r)}$, it is difficult to compute the global optimization. We would like to optimize $\mathbf{U}^{(r)}$, $\mathbf{V}^{(r)}$ and b_r alternatively as the procedure of 2DLDA [19]. More concretely, we fix one parameter and optimize the other one.

1) Fixing $\mathbf{V}^{(r)}$ and Optimizing $\mathbf{U}^{(r)}$ and b_r: As seen from Eq. (4) and Eq. (8), we reformulate the optimization problem in Eq. (10) as follows

$$\begin{aligned}
&\arg\min\mathcal{L}\left(\mathbf{U}^{(r)}, \mathbf{V}^{(r)}, b_r\right)\\
&= \sum_{i=1}^{l}\left(\sum_{j=1}^{k}\left(\mathbf{u}_j^{(r)}\right)^T\mathbf{X}_i\mathbf{v}_j^{(r)} + b_r - y_{ir}\right)^2\\
&\quad +\alpha\sum_{j=1}^{k}Tr\left(\mathbf{u}_j^{(r)}\left(\mathbf{v}_j^{(r)}\right)^T\mathbf{v}_j^{(r)}\left(\mathbf{u}_j^{(r)}\right)^T\right).
\end{aligned} \tag{11}$$

When $\mathbf{V}^{(r)}$ is fixed, we need to compute a set of $\mathbf{u}_j^{(r)}$ for $j = 1, 2, \ldots, k$. It seems difficult to compute them simultaneously.

Nevertheless, if we reformulate the problem in Eq. (11), its closed solution can be derived directly.

Denote

$$\mathbf{F}_i^{(r)} = \begin{bmatrix} \mathbf{X}_i\mathbf{v}_1^{(r)} \\ \mathbf{X}_i\mathbf{v}_2^{(r)} \\ \vdots \\ \mathbf{X}_i\mathbf{v}_k^{(r)} \end{bmatrix}_{mk\times 1}, \hat{\mathbf{U}}^{(r)} = \begin{bmatrix} \mathbf{u}_1^{(r)} \\ \mathbf{u}_2^{(r)} \\ \vdots \\ \mathbf{u}_k^{(r)} \end{bmatrix}_{mk\times 1},$$

$$\mathbf{D}^{(r)} = \begin{bmatrix} \left(\mathbf{v}_1^{(r)}\right)^T \mathbf{v}_1^{(r)}\mathbf{I}_{m\times m} & & \\ & \cdots & \\ & & \left(\mathbf{v}_k^{(r)}\right)^T \mathbf{v}_k^{(r)}\mathbf{I}_{m\times m} \end{bmatrix}. \quad (12)$$

Then, Eq. (11) is equivalent to

$$\arg\min \mathcal{L}\left(\hat{\mathbf{U}}^{(r)}, b_r\right) = \sum_{i=1}^{l}\left(\left(\hat{\mathbf{U}}^{(r)}\right)^T \mathbf{F}_i^{(r)} + b_r - y_{ir}\right)^2 + \alpha\left(\hat{\mathbf{U}}^{(r)}\right)^T \mathbf{D}^{(r)}\mathbf{U}^{(r)}. \quad (13)$$

Denote

$$\mathbf{F}^{(r)} = [\mathbf{F}_1^{(r)}, \mathbf{F}_2^{(r)}, \ldots, \mathbf{F}_l^{(r)}]_{mk\times l},$$
$$\mathbf{y}^{(r)} = [y_{1r}, y_{2r}, \ldots, y_{lr}]_{1\times l}. \quad (14)$$

Then, Eq. (13) becomes

$$\arg\min \mathcal{L}\left(\hat{\mathbf{U}}^{(r)}, b_r\right) = \left(\left(\hat{\mathbf{U}}^{(r)}\right)^T \mathbf{F}^{(r)} + b_r\mathbf{e} - \mathbf{y}^{(r)}\right) \times \left(\left(\hat{\mathbf{U}}^{(r)}\right)^T \mathbf{F}^{(r)} + b_r\mathbf{e} - \mathbf{y}^{(r)}\right)^T + \alpha\left(\hat{\mathbf{U}}^{(r)}\right)^T \mathbf{D}^{(r)}\mathbf{U}^{(r)}. \quad (15)$$

Take the derivative of $\mathcal{L}(\hat{\mathbf{U}}^{(r)}, b_r)$ with respect to $\hat{\mathbf{U}}^{(r)}$, b_r and set them to zeros, we will have the optimal solutions to the problem in Eq. (11) as follows

$$\hat{\mathbf{U}}^{(r)} = \left[\mathbf{F}^{(r)}\mathbf{L}_c\left(\mathbf{F}^{(r)}\right)^T + \alpha\mathbf{D}^{(r)}\right]^{-1}\mathbf{F}^{(r)}\mathbf{L}_c\left(\mathbf{y}^{(r)}\right)^T,$$
$$b_r = \frac{1}{l}\left(\mathbf{y}^{(r)} - \left(\hat{\mathbf{U}}^{(r)}\right)^T \mathbf{F}^{(r)}\right)\mathbf{e}^T \quad (16)$$

where $\mathbf{L}_c = \mathbf{I} - \frac{1}{l}\mathbf{e}^T\mathbf{e}$ as defined in Table I.

In summary, when $\{\mathbf{v}_i^{(r)}\}_{i=1}^k$ is fixed, the solution of MRR can be computed by Eq. (16) directly, without iteration. In Section IV (A), we will show that when $\{\mathbf{v}_i^{(r)}\}_{i=1}^k$ is fixed, our derived results are the global solutions to the problem in Eq. (11).

2) Fixing $\mathbf{U}^{(r)}$ and Optimizing $\mathbf{V}^{(r)}$ and b_r: When $\mathbf{U}^{(r)}$ is fixed, the optimal $\mathbf{V}^{(r)}$ and b_r to the problem in Eq. (10) can also be derived in a closed form.

Denote

$$\mathbf{G}_i^{(r)} = \begin{bmatrix} \mathbf{X}_i^T\mathbf{u}_1^{(r)} \\ \mathbf{X}_i^T\mathbf{u}_2^{(r)} \\ \vdots \\ \mathbf{X}_i^T\mathbf{u}_k^{(r)} \end{bmatrix}_{nk\times 1}, \hat{\mathbf{V}}^{(r)} = \begin{bmatrix} \mathbf{v}_1^{(r)} \\ \mathbf{v}_2^{(r)} \\ \vdots \\ \mathbf{v}_k^{(r)} \end{bmatrix}_{nk\times 1},$$
$$\mathbf{G}^{(r)} = [\mathbf{G}_1^{(r)}, \mathbf{G}_2^{(r)}, \ldots, \mathbf{G}_l^{(r)}]_{nk\times l}.$$
$$\mathbf{Q}^{(r)} = \begin{bmatrix} \left(\mathbf{u}_1^{(r)}\right)^T \mathbf{u}_1^{(r)}\mathbf{I}_{n\times n} & & \\ & \cdots & \\ & & \left(\mathbf{u}_k^{(r)}\right)^T \mathbf{u}_k^{(r)}\mathbf{I}_{n\times n} \end{bmatrix}. \quad (17)$$

Then Eq. (10) becomes

$$\arg\min \mathcal{L}\left(\hat{\mathbf{V}}^{(r)}, b_r\right) = \left(\left(\hat{\mathbf{V}}^{(r)}\right)^T \mathbf{G}^{(r)} + b_r\mathbf{e} - \mathbf{y}^{(r)}\right) \times \left(\left(\hat{\mathbf{V}}^{(r)}\right)^T \mathbf{G}^{(r)} + b_r\mathbf{e} - \mathbf{y}^{(r)}\right)^T + \alpha\left(\hat{\mathbf{V}}^{(r)}\right)^T \mathbf{Q}^{(r)}\hat{\mathbf{V}}^{(r)}. \quad (18)$$

Similarly, take the derivative of $\mathcal{L}(\hat{\mathbf{V}}^{(r)}, b_r)$ with respect to $\hat{\mathbf{V}}^{(r)}$, b_r and set them to zeros, we will derive the optimal solutions to the problem in Eq. (10) as follows:

$$\hat{\mathbf{V}}^{(r)} = \left[\mathbf{G}^{(r)}\mathbf{L}_c\left(\mathbf{G}^{(r)}\right)^T + \alpha\mathbf{Q}^{(r)}\right]^{-1}\mathbf{G}^{(r)}\mathbf{L}_c\left(\mathbf{y}^{(r)}\right)^T,$$
$$b_r = \frac{1}{l}\left(\mathbf{y}^{(r)} - \left(\hat{\mathbf{V}}^{(r)}\right)^T \mathbf{G}^{(r)}\right)\mathbf{e}^T. \quad (19)$$

In conclusion, when we fix one parameter and optimize the other, we can derive the solutions in a closed form.

The above process aims to train classifiers for the r-th category. As mentioned above, we can train c classifiers, i.e. $\{(\hat{\mathbf{U}}^{(r)}, \hat{\mathbf{V}}^{(r)}, b_r)\}_{r=1}^c$, for c categories in the similar way. When testing point $\mathbf{X}_{l+1}$ comes, we can directly compute the corresponding regression values and choose the one that is the largest. Then we assign this point to the selected category. More concretely, we first compute the c regression values of $\mathbf{X}_{l+1}$. They are $\sum_{j=1}^k (\mathbf{u}_j^{(r)})^T\mathbf{X}_{l+1}\mathbf{v}_j^{(r)} + b_r$ for $r = 1, 2, \ldots, c$. Then $\mathbf{X}_{l+1}$ is categorized to the p-th class, where $p = \arg\max_r(\sum_{j=1}^k (\mathbf{u}_j^{(r)})^T\mathbf{X}_{l+1}\mathbf{v}_j^{(r)} + b_r)$. The reason is that we use the same way in designing the label vector for training data, i.e., $\mathbf{y}^{(r)}$ in Eq. (9). Besides, MRR can also be regarded as reducing the dimensionality of original data to a c-dimensional subspace, in which the label information is revealed.

Finally, there are several points should be highlighted here.

1) The first one is about initialization. As seen from above procedure, MRR is solved in an iterative way. We would like to initialize $\mathbf{V}$ by the same way as in [19]. More concretely, the initialization is $[\mathbf{I}_{k\times k}, \mathbf{0}_{k\times(n-k)}]^T$. As stated in [19], this kind of initialization performs well. Our experimental results also validate the effectiveness of this initialization. In the following experiments, we use this kind of initialization if there is no specification. In next section, we will also show some other kinds of initialization. Some experimental results are also provided for comparison.

TABLE II
PROCEDURE OF MRR

Training Process:
Input: Training matrix data set: $\{\mathbf{X}_i \mid i = 1, 2, \ldots, l\}$, label vectors: $\{\mathbf{y}_i \mid i = 1, 2, \ldots, l\}$, balance parameter α.
Output: Projections for c classes: $\hat{\mathbf{U}}^{(r)}$, $\hat{\mathbf{V}}^{(r)}$, b_r, $r = 1, 2, \ldots, c$.
For $r = 1, 2, \ldots, c$
1. Initialize $\mathbf{v}_j^{(r)} = \left[0 \cdots 0 \underset{j-\text{th}}{1} 0 \cdots 0 \right]$ and formulate $\mathbf{F}_i^{(r)}$ in Eq. (12);
2. Alternatively update $\hat{\mathbf{U}}^{(r)}$ and $\hat{\mathbf{V}}^{(r)}$ until convergence.
a. Update $\hat{\mathbf{U}}^{(r)}$ and b_r using Eq. (16), where $\hat{\mathbf{v}}_j^{(r)}$ is derived in the former step. $\hat{\mathbf{F}}_i^{(r)}$ and $\mathbf{D}^{(r)}$ in Eq. (12) are computed based on the latest $\hat{\mathbf{V}}^{(r)}$;
b. Update $\hat{\mathbf{V}}^{(r)}$ and b_r using Eq. (19), where $\hat{\mathbf{u}}_j^{(r)}$ is derived in the former step. $\hat{\mathbf{G}}_i^{(r)}$ and $\mathbf{Q}^{(r)}$ in Eq. (17) are computed based on the latest $\hat{\mathbf{U}}^{(r)}$.
Testing Process:
Input: Testing matrix data set: $\{\mathbf{X}_i \mid i = l+1, l+2, \ldots, l+t\}$. The projectors for c classes: $\hat{\mathbf{U}}^{(r)}$, $\hat{\mathbf{V}}^{(r)}$ and b_r for $r = 1, 2, \ldots, c$.
Output: The labels of testing data: $\{\mathbf{y}_i \mid i = l+1, l+2, \ldots, l+t\}$.
1. Compute the c regression values of $\mathbf{X}_i$: $\{\sum_{j=1}^{k} (\mathbf{u}_j^{(r)})^T \mathbf{X}_i \mathbf{v}_j^{(r)} + b_r\}_{r=1}^{c}$ for $i = l+1, l+2, \ldots, l+t$.
2. Assign the label of $\mathbf{X}_i$, i.e., $\mathbf{y}_i$ for $i = l+1, l+2, \ldots, l+t$. $y_{ij} = \begin{cases} 1, & j = \arg\max_r (\sum_{j=1}^{k} (\mathbf{u}_j^{(r)})^T \mathbf{X}_i \mathbf{v}_j^{(r)} + b_r) \\ 0, & \text{otherwise} \end{cases}$

2) The second one is about the convergence behavior of above iteration. In next section, we will show that through this kind of iteration, the objective function in Eq. (9) will decrease. In other words, the above iteration is convergent. Our experiments also show that this kind of iteration converges fast. It is often less than ten times.
3) Although the computed projecting vectors are $\hat{\mathbf{U}}^{(r)}$ and $\hat{\mathbf{V}}^{(r)}$, not $\mathbf{U}^{(r)}$ and $\mathbf{V}^{(r)}$, they are just different arrangements of the same vectors, i.e., $\{\mathbf{u}_j^{(r)}\}_{j=1}^{k}$ and $\{\mathbf{v}_j^{(r)}\}_{j=1}^{k}$.

In summary, the procedure of MRR is listed in Table II.

IV. Performance Analysis

In this section, we will analyze the performance of MRR in four aspects, i.e., convergence behavior, initialization, computational complexity analysis and parameter determination.

A. Convergence Analysis

As mentioned above, for each category, MRR is solved in an alternative way. Namely, we fix one variable and compute the other. The following proposition shows that when $\mathbf{V}^{(r)}$ is fixed, the results in Eq. (16) are the global solutions to the problem in Eq. (10). Similarly, the results in Eq. (19) are also the solutions to the problem in Eq. (10) if $\mathbf{U}^{(r)}$ is fixed.

Proposition 1: When $\mathbf{V}^{(r)}$ is fixed, the results $\mathbf{U}^{(r)}$, derived from $\hat{\mathbf{U}}^{(r)}$, and b_r in Eq. (16) are the global solutions to the problem in Eq. (10). Similarly, when $\mathbf{U}^{(r)}$ is fixed, the results derived from Eq. (19) are also the global solutions to the problem in Eq. (10).

See the proof in the Appendix. Intuitively, when an image is compressed into one direction (row or column) by multiplying several regress vectors, our method could find the best corresponding vectors which can regress the compressed image to its label vector.

The above proposition shows that our algorithm could find the global optimization in each step. The following proposition shows that our algorithm will monotonically decrease the objective function of the problem in Eq. (9) in each iteration.

Proposition 2: The procedures of MRR will monotonically decrease the objective function of the problem in Eq. (9) in each iteration.

See the proof in the Appendix. Intuitively, by optimizing the above problem alternatively, we can find the vectors that can regress each image to its label with more and more smaller regression errors. The final regression vectors can be regarded as the classifier which can be used for the classification of new images.

There are two points should be mentioned here. (1) In the above deduction, in fact, b_r is needed to update only one time in each iteration since the derivations of $\hat{\mathbf{U}}^{(r)}$ and $\hat{\mathbf{V}}^{(r)}$ are not related to b_r. We have not distinguished it in deriving $\hat{\mathbf{U}}^{(r)}$ and $\hat{\mathbf{V}}^{(r)}$. (2) The final results of MRR are closely related to the initialization. We use the same strategy as in 2DLDA. Experimental results show that this kind of initialization works well. Therefore, our derived solution may be close to the global optimal solution.

B. Initialization

Since our method, together with other two methods, i.e., 2DLDA and GBR, are solved in an alternative way, we would like to discuss their initialization problem. There are totally three different kinds of initial strategies that we will introduce.

The first kind of initialization, which has been mentioned above, is empirical. As shown in [19], the initialization of 2DLDA is $\mathbf{R} = [\mathbf{I}_{l_2 \times l_2}, \mathbf{0}_{l_2 \times (n-l_2)}]^T$, where l_2 is the second reduced dimensionality. Similarly, we can initialize GBR by $[1, \mathbf{0}_{1 \times (n-1)}]^T$ and MRR by $[\mathbf{I}_{k \times k}, \mathbf{0}_{k \times (n-k)}]^T$. As stated in [19] and seen from the following results, although this kind of initialization is simple, it has been proved to be effective.

The second kind of initialization is random. We generate some random variables, which is sampled from a uniform distribution (range from 0 to 1) or a standard norm distribution to form the matrixes with corresponding sizes. Then, the column vectors of generated initialization matrix of GBR and MRR is changed to be orthogonal. Intuitively, since each column vector of MRR initialization corresponds to the weight of column vectors of $\mathbf{X}$ in following regression, we want these column vectors to be orthogonal.

Finally, we use the bootstrap technology [26] to generate each column vector for initialization. More concretely, for each column, we randomly (uniformly) selected several rows, whose values are 1. The others are 0. Different from the above two strategies, whose elements are decimal, these initialization matrixes are binary. The reason why we do this is evoked by the first kind of strategy. In the first strategy, only a few columns of $\mathbf{X}$ are selected. For example, $[\mathbf{I}_{k \times k}, \mathbf{0}_{k \times (n-k)}]^T$ in

MRR means that only the first k columns are selected for next regression. If we use bootstrap, we can employ multiple columns of $\mathbf{X}$ simultaneously.

In Section V (C), we will provide some experimental results to compare different initializations.

C. Computational Complexity Comparison

Since one of our motivations is to reduce the computational complexity of traditional regression methods, we would like to analyze the computational complexity of different related methods, including Tikhonov regularized least square regression (1DREG), LDA, 2DLDA, GBR and MRR. Since different implementations of these methods may have different computational complexities, we would like to give common analyses merely.

The first group of methods is LDA and 2DLDA. As seen from the procedure of LDA and 2DLDA, the most time consuming step is to solve the eigen-decomposition problem. Its computational complexity is about $O(D^3)$, where D is the dimensionality of original data. Thus, traditional LDA has the computational complexity $O(m^3n^3)$. 2DLDA solves two eigen-decomposition problem with the sizes m and n respectively. Thus, its computational complexity is $O(s(m^3 + n^3))$, where s is the time of iteration.

The second group of methods is 1DREG, GBR and MRR. The most time consuming step of 1DREG, GBR and MRR is to solve the regularized least square regression problem. It has the computational complexity about $O(D^2)$, where D is also the dimensionality. Since 1DREG treats a $m \times n$ matrix as a mn-dimensional vector, its computational complexity is $O(c(m^2n^2))$. In each iteration, GBR solves two regularized least square regression problems with the data dimensionality m and n respectively. Assume that there are mainly s iterations, the computational complexity of GBR is $O(sc(m^2+n^2))$. Similar to GBR, MRR solves two regularized least square regression problems with the dimensionality mk and nk respectively. Thus, its computational complexity is $O(sc(m^2 + n^2)k^2)$. Commonly, s is less than 10 and k is far less than $\min\{m, n\}$. (We will discuss its determination and it is set to 2 in the following experiments.). Thus, the computational complexity of MRR is similar to GBR and much smaller than 1DREG.

In summary, the computational complexities of four methods have the following relationships. GBR $\leq$ MRR $<$ 2DLDA $<$ 1DREG $<$ LDA. Nevertheless, different implementations of the same method may cost different time in real computing. We will give some numerical results in next section.

D. Parameter Determination

There are mainly two important parameters in our MRR algorithm. The first one is regularization parameter α and the second is the rank of regression, i.e., k. As stated in [27], parameter determination is still an open problem in the related fields. Thus, we determine parameters heuristically and empirically.

For the first parameter α, it is used to balance the influence of two terms, i.e., the regression and regularization. When α is large, we will neglect the requirement of regression. When it is small, the problem of over fitting will appear. In this paper, we determine it by five fold cross validation [28]. The experiments in Section V(E) will show numerical results with different α. See more details there.

The second parameter is the rank of regression, i.e., k. Obviously, this parameter balances the effects of fitting error and computational complexity. The larger k is, the smaller fitting error is and the larger computational complexity is. More importantly, the following proposition indicates that k is also a parameter to balance the capacity for learning and generalization essentially. We would like to give the explanations after showing this result.

As seen from Eq. (7), since the counterpart of $\mathbf{w}^{(r)}$ is $\mathbf{U}^{(r)}(\mathbf{V}^{(r)})^T$, we would like to reveal their relationship in following proposition.

Proposition 3: Assume $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ are any k vectors of dimensionality m, $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ are k vectors of dimensionality n. If $k = \min(m, n)$, then the dimensionality of space spanned by $(Vec(\sum_{i=1}^{k} \mathbf{u}_i(\mathbf{v}_i)^T))^T Vec(\mathbf{X}_i)$ is mn. Here $Vec(\cdot)$ represents the vectorization of a matrix.

See the proof in the Appendix. We would like to explain previous discussion now. When $k = \min(m, n)$ and we aim at minimizing $Tr((\mathbf{U}^{(r)})^T\mathbf{X}_i\mathbf{V}^{(r)})$, the above proposition indicates that the searching space of $\mathbf{U}^{(r)}(\mathbf{V}^{(r)})^T$ is very similar to that in minimizing $(Vec(\sum_{i=1}^{k} \mathbf{u}_i(\mathbf{v}_i)^T))^T Vec(\mathbf{X}_i)$. In other words, $(Vec(\sum_{i=1}^{k} \mathbf{u}_i(\mathbf{v}_i)^T))^T Vec(\mathbf{X}_i)$ and $Tr(\mathbf{U}^{(r)}\mathbf{X}_i\mathbf{V}^{(r)})$ are more likely to be the same. That is to say, the first objective function of MRR is close to the corresponding objective of 1DREG. On one hand, when a matrix is expanded as a vector, the regression model has more free parameters ($m \times n$). It has smaller fitting error. Nevertheless, it tends to be over fitting. On the other hand, when $k = 1$, we only have $m + n$ free parameters and the regression error is large. Thus, k can be regarded as a parameter to balance the fitting error and the extent in avoiding over fitting. Since fitting error can measure the capacity of learning and the extent in avoiding over fitting can measure the capacity of generalization, k is a parameter to balance these two capacities in essence. Comparing MRR with 1DREG and GBR, we know that 1DREG has the largest k. Thus, it has smallest fitting error, which indicates its strongest learning capacity. Nevertheless, it trends to be over fitting, which means that it has the weakest capacity for generalization. On the contrary, GBR has the strongest capacity for generalization but weakest capacity for learning, since it assumes $k = 1$. Recalling the basic rule in learning theory, these two capacities can not be improved simultaneously for a learning algorithm [5]. Thus, MRR can be regarded as a general tradeoff between these two capacities. That is to say, our method is the tradeoff of advantages of GBR (strongest capacity for generalization) and 1DREG (strongest capacity of learning) and performs better than both of them.

From the view of image processing, k can be regarded as the parameter to measure the dependence of matrix elements in regression. 1DREG treats each element equally since it has $m \times n$ free parameters. GBR, however, does not treat each element of a matrix equally. For instance, when we multiple a vector on the left of a matrix, each column, not each element,

Fig. 2. Then sample images from six data sets. From top to bottom: Umist, Coil, Usps, AR, Pie, and Mnist.

is treated equally. Similarly, each row is treated equally when a matrix times a vector. In other words, when k is small, it adds too strict constraints on the correlations among different elements of a matrix. When k is too large, it neglects their correlations in an image.

One point should be explained here. Compared with GBR, who assumes that $k = 1$, the increase of rank will lead to the over fitting problem and increase the computational complexity. Nevertheless, we have set $k = 2$ and added a regularizer shown in Eq. (8) to avoid over fitting. Besides, experimental results show that $k = 2$ is better enough. That is to say, the time addition is also limited. On the contrary, MRR has much smaller computational complexity than 1DREG. The following experimental results in Section V(E) show that the classification accuracy does not change drastically when $k \geq 2$. Thus, we set $k = 2$ in the following experiments.

V. Experiments

There are mainly six kinds of experiments in this section. The first group mainly focuses on the evaluation of classification accuracy with K-nearest neighborhood classifier (KNN). To demonstrate whether the out-performance of our method is dominated by the following classifier, we provide some classification results with Support Vector Machine (SVM) in the second group. The third group contains experiments with different initialization and the forth group consists of computational time comparison results. Finally, classification accuracies with different parameters are presented. We would like to introduce data sets and evaluation methods at first.

A. Data Description and Evaluation Metric

There are mainly six different kinds of matrix data sets, i.e., Umist,[1] Coil,[2] Usps,[3] Ar,[4] Pie,[5] and Mnist.[6] They can be classified into three types. Umist, Ar and Pie are face images. Coil data set contains images of 20 different objects. Usps and Mnist contain images of handwritten digits. The data size ranges from about 500 to 11 000 and the image resolution ranges from 16×16 to 64×64. In other words, the dimensionality of vectorized sample ranges from about 200 to 4000. For illustration, we select ten sample images per data set and rescale them to 16×16. They are shown in Fig. 2. Additionally, after some preprocessing, the detailed statistical characters of different data sets are listed in Table III.

TABLE III
Characters of Different Data Sets

Data	Size ($l + t$)	Scale ($m \times n$)	Class	No. of Training
Umist	575	28×23	20	$40, 60, \ldots, 220$
Coil	1440	32×32	20	$40, 60, \ldots, 220$
Usps	2007	16×16	10	$20, 30, \ldots, 110$
Ar	140	64×64	10	$20, 30, \ldots, 110$
Pie	11554	32×32	68	$136, 204, \ldots, 748$
Mnist	5000	28×28	10	$20, 30, \ldots, 110$

As mentioned above, since 1DREG, GBR and 2DLDA is closely related to MRR, we would like to compare the performance of MRR with them. Besides, we also provide the results of LDA since 2DLDA can be regarded as its two dimensional extension. Nevertheless, since LDA and 2DLDA can not be directly used for classification, we would like to use them for projection and then employ other classifiers, e.g., KNN or SVM, for classification. For justice, we also regard 1DREG and MRR as the methods to project the matrix data into a c-dimensional subspace and then employ the same classifier for classification, although they can be used for classification directly. The results of these classifiers are also provided as the baseline. Since feature based methods focus on description of image appearance and our work aims at developing a new learning method, we have not compared with feature based methods.

B. Classification Accuracy Comparison

In this subsection, we provide some classification results with different classifiers on different data sets. In the first group, we use KNN ($K = 1$) as the classifier. By randomly splitting the original data into training set and testing set for 50 independent runs, we select $2, 3, \ldots, 11$ samples from each category and the other is set as testing images. As in traditional applications, the dimensionality of LDA and 2DLDA projections is $c - 1$. 1DREG, GBR and MRR assume that the low dimensionality is c. The regularized parameter α is determined by five folder cross validation.

With different data sets and different numbers of training samples, the mean classification accuracies of 50 independent runs on all data sets are shown in Fig. 3(a) (Umist data), Fig. 3(b) (Coil data), Fig. 3(c) (Usps data), Fig. 3(d) (Ar data), Fig. 3(e) (Pie data) and Fig. 3(f) (Mnist data), respectively.

There are several observations from the performance comparisons as follows.

1) Among different methods and different data sets, MRR performs best. It achieves the highest accuracy in most cases, especially on the data sets such as Coil and Usps. This is mainly due to the fact that MRR has smaller fitting error and stronger capacity for generalization.

[1] http://images.ee.umist.ac.uk/danny/database.html.
[2] http://www1.cs.columbia.edu/CAVE/research/softlib/coil-20.html.
[3] http://www-i6.informatik.rwth-aachen.de/ keysers/usps.html.
[4] http://www2.ece.ohio-state.edu/ aleix/ARdatabase.html.
[5] http://www.zjucadcg.cn/dengcai/Data/FaceData.html.
[6] http://yann.lecun.com/exdb/mnist/.

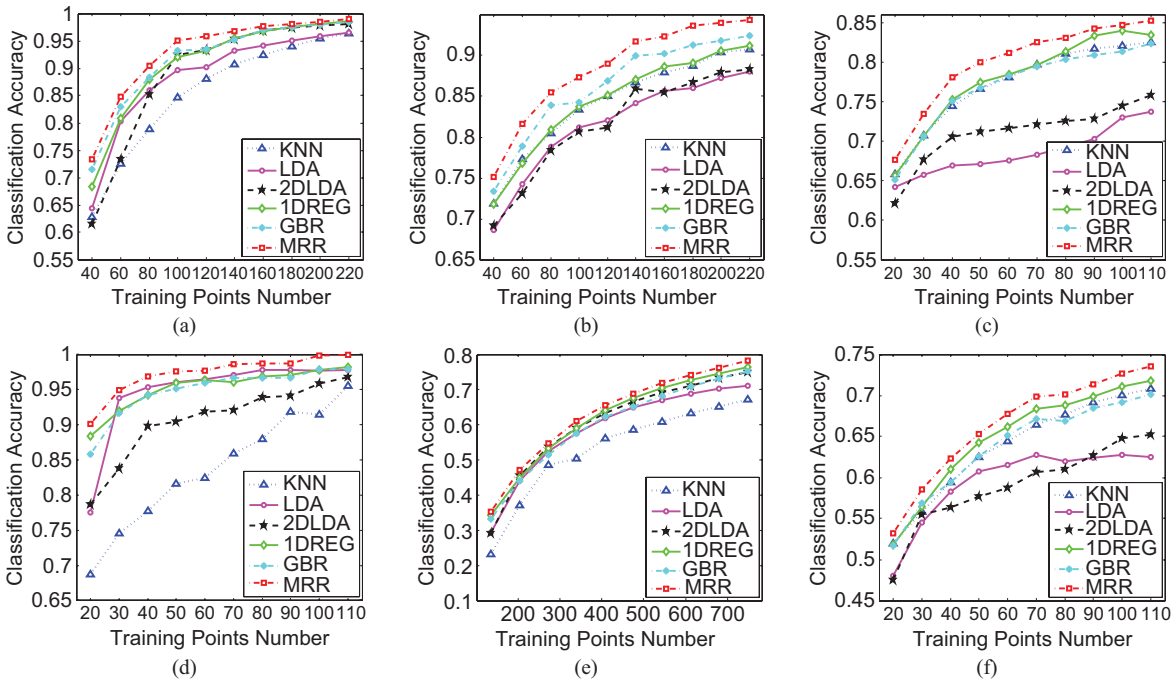

Fig. 3. Classification results of different methods on six different data sets with different numbers of training points. (a) Results on the Umist data. (b) Results on the Coil data. (c) Results on the Usps data. (d) Results on the AR data. (e) Results on the Pie data. (f) Results on the Mnist data.

TABLE IV

CLASSIFICATION ACCURACY OF DIFFERENT METHODS ON UMIST DATA BY USING SVM. (MEAN ± STD/%)

	SVM	LDA	2DLDA	1DREG	GBR	MRR
40	59.82±1.47	63.66±1.87	61.74±1.69	63.85±1.84	65.59±1.54	**69.20**±1.30
60	66.65±1.74	75.33±1.85	67.83±1.66	74.34±1.74	75.73±1.81	**78.81**±1.40
80	72.73±1.71	82.76±1.58	82.17±1.15	82.46±1.64	82.64±1.37	**85.19**±1.14
100	77.38±1.44	86.37±0.95	**87.66**±1.68	85.63±1.34	86.41±1.44	**88.63**±1.12
120	81.01±1.38	88.10±0.96	87.86±0.96	87.76±1.00	87.73±1.26	**90.56**±0.98
140	85.03±1.42	**91.33**±0.95	90.03±0.99	90.29±0.97	89.40±0.90	**92.40**±0.83
160	87.78±0.81	91.58±0.79	**92.70**±0.89	91.90±0.82	91.07±0.98	**93.78**±0.76
180	89.81±1.07	92.59±0.69	93.03±0.63	92.83±0.73	92.33±0.94	**94.73**±0.74
200	91.25±0.90	92.49±0.82	93.62±0.68	93.52±0.69	92.81±0.91	**95.51**±0.72
220	92.39±0.76	93.23±0.87	93.49±0.75	94.23±0.77	93.45±0.86	**96.21**±0.69

2) With the increase of training points' number, all methods achieve higher accuracies. This is consistent with intuition since we have more information for training.
3) For classification, 2D based methods do not always perform better than 1D based methods. Take the results in Fig. 3(d) as an example, LDA achieves higher accuracy than 2DLDA in most cases. The reason may be that the adding constraints in 2DLDA will degrade the performances.
4) When we represent the original data by its low dimensional embedding, it does not always helpful for the next classification. Take Fig. 3(f) as an example, comparing with the results in using KNN to classify the embedding derived by LDA and 2DLDA, we achieve higher accuracy when KNN is used to classify the original data directly.

Different from above settings, in the second group of experiments, we use another classifier SVM for classification. These experiments are proposed to demonstrate that the outperformance of MRR is not dominated by the using of KNN classifier. We choose three data sets, Umist, Coil and Mnist for illustration. The other settings are the same as previous and the mean and standard derivation values are shown in Table IV, Table V and Table VI. In statistical view, the largest values are boldfaced.

As seen from Tables IV-VI, we can draw almost the same conclusion as previous. Concretely, MRR also achieves the highest accuracies in most cases. In other words, the outperformance of MRR is not dominated by the following classifier. More interestingly, in some data sets, such as Coil, LDA based method may perform better than other regression based methods. In other cases, however, regression methods achieve higher accuracy. This may be caused by the fact that different data have different structures and different methods could characterize part of them.

C. Different Initialization

In this section, we would like to show the influence of different initializations. As mentioned in Section IV(B), we

TABLE V
CLASSIFICATION ACCURACY OF DIFFERENT METHODS ON COIL DATA BY USING SVM. (MEAN ± STD/%)

	SVM	LDA	2DLDA	1DREG	GBR	MRR
40	70.16±1.27	69.31±1.25	53.11±1.59	70.68±1.03	70.54±1.20	**73.86**±1.02
60	74.95±1.20	75.16±1.20	59.75±1.44	74.34±1.21	75.55±1.28	**79.01**±1.12
80	76.99±1.07	78.78±1.17	62.41±1.78	78.45±0.99	78.10±1.32	**81.52**±1.01
100	80.50±1.24	80.81±1.02	67.00±1.89	80.45±1.03	81.45±1.02	**83.46**±0.85
120	82.73±0.96	81.97±1.02	69.83±1.83	82.78±0.93	82.48±0.93	**85.78**±0.83
140	83.42±0.88	82.61±0.83	70.36±1.02	83.43±0.64	83.39±0.90	**86.91**±0.62
160	85.46±0.84	83.35±0.80	71.04±1.09	84.24±0.66	85.95±0.91	**87.62**±0.65
180	86.07±0.81	86.40±0.74	75.99±1.59	87.20±0.68	86.70±0.68	**89.17**±0.61
200	87.54±0.68	87.48±0.84	77.41±0.86	**88.61**±0.56	87.31±0.84	**89.99**±0.66
220	88.84±0.70	88.39±0.62	81.56±0.98	**89.32**±0.72	87.63±0.73	**90.24**±0.60

TABLE VI
CLASSIFICATION ACCURACY OF DIFFERENT METHODS ON MNIST DATA BY USING SVM. (MEAN ± STD/%)

	SVM	LDA	2DLDA	1DREG	GBR	MRR
20	**52.06**±1.58	48.21±1.83	**50.39**±1.82	**51.17**±1.67	**50.71**±1.27	**52.22**±1.40
30	**57.12**±1.32	55.98±1.48	52.67±1.47	**56.99**±1.30	55.03±1.64	**58.46**±1.36
40	**61.56**±1.43	59.23±1.66	58.36±1.22	**61.17**±1.09	58.60±1.39	**62.30**±1.31
50	**63.99**±1.26	60.50±1.13	58.68±1.47	63.49±1.23	61.36±1.30	**66.05**±1.09
60	**66.60**±0.99	62.38±1.19	59.68±1.96	66.26±1.02	63.04±1.30	**68.52**±0.98
70	67.48±0.99	62.30±1.13	59.74±1.41	67.40±0.89	64.26±1.06	**70.14**±0.86
80	69.32±1.00	62.70±1.25	61.12±1.49	68.92±0.93	65.72±1.05	**72.24**±0.84
90	70.09±0.81	62.64±1.03	62.85±1.83	70.09±0.80	66.31±1.01	**73.91**±0.95
100	71.71±0.88	63.41±0.96	63.35±1.79	71.17±0.99	67.66±0.96	**75.40**±0.85
110	72.15±0.86	63.54±0.85	64.64±1.68	71.84±0.78	67.99±1.10	**75.77**±0.75

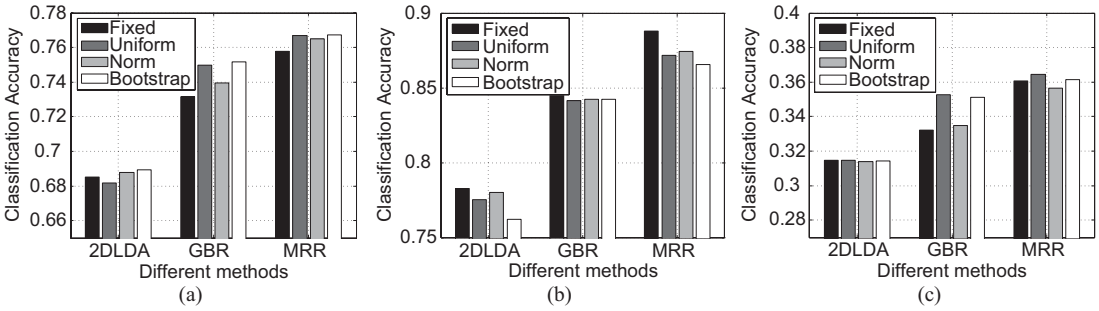

Fig. 4. Classification results of different methods with different kinds of initialization. (a) Coil data. (b) Ar data. (c) Pie data.

have discussed three kinds of initializations. Briefly, the first kind uses fixed value, we name it as 'Fixed' in the following experiments. The second kind of methods initializes randomly with either uniform or norm distributions. We name it as 'Unified' and 'Norm'. Finally, when the bootstrap technology is employed, we name it as 'Bootstrap'.

We have conducted experiments on three data sets, i.e., Coil, Ar and Pie. With fixed number of training points (2 for each category), we randomly split the original data set as training set and testing set for 50 runs. We compare MRR with 2DLDA and GBR since they are all solved in an alternative way. We use KNN as the classifier and the other settings are the same as previous. All the experimental results, which are the average classification accuracies of 50 runs, are shown in Fig. 4.

As seen from the results in Fig. 4, we can conclude that three methods have different performances with different initializations. It is mainly due to the fact that our solving strategy only finds the local optimization. Different initializations imply different local optimal solutions and different classification results. Nevertheless, the variance between different initialization is not so significant. Thus, we have used the first kind of initialization in other experiments. Besides, the effectiveness of initialization takes different influence on three methods on different data sets. For example, the variance of 2DLDA is largest on Ar data set and smallest on the Pie data. Finally, as seen from the results in Fig. 4, MRR achieved the highest classification accuracy in almost all cases.

D. Computational Time Comparison

Another motivation for developing our algorithm is to reduce the computational complexity of original one

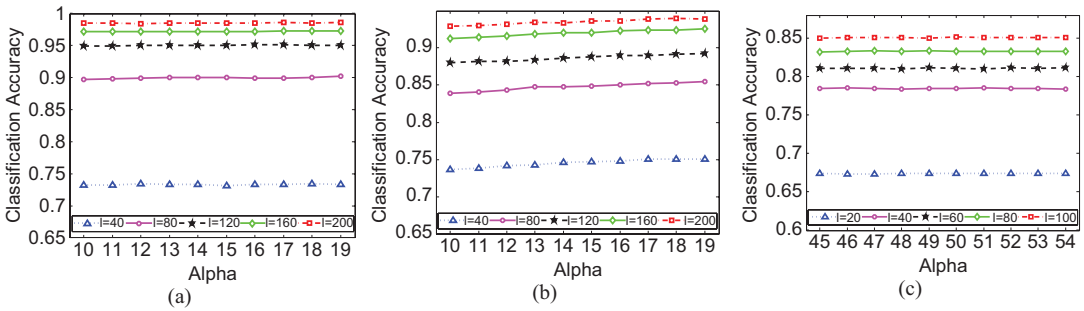

Fig. 5. Classification results of MRR with different parameter α. (a) Umist data. (b) Coil data. (c) Usps data.

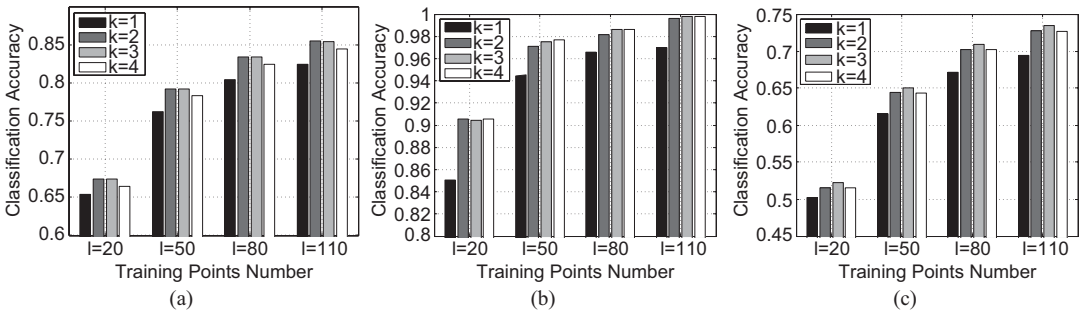

Fig. 6. Classification results of MRR with different rank of regression, i.e., k. (a) Usps data. (b) Ar data. (c) Mnist data.

TABLE VII
COMPUTATIONAL TIME ON AR DATA WITH DIFFERENT NUMBERS OF TRAINING POINTS. THE DATA SCALE IS 64×64. (MEAN $\pm$ STD)

	LDA	2DLDA	1DREG	GBR	MRR
20	215.79±2.38	6.71 ±1.73	63.78±0.89	0.04±0.02	0.12±0.02
40	217.76±2.40	8.31 ±1.62	64.42±0.89	0.07±0.02	0.15±0.03
60	219.96±2.59	10.75±1.94	65.60±0.72	0.14±0.19	0.15±0.06
80	225.12±2.80	13.67±1.49	65.92±0.46	0.15±0.17	0.25±0.16
100	240.75±2.90	14.82±1.02	66.82±0.28	0.20±0.09	0.28±0.11

TABLE VIII
COMPUTATIONAL TIME ON PIE DATA WITH DIFFERENT NUMBERS OF TRAINING POINTS. THE DATA SCALE IS 32×32. (MEAN $\pm$ STD)

	LDA	2DLDA	1DREG	GBR	MRR
68	9.71±1.88	2.03±1.13	6.92 ±0.89	1.06±0.25	1.34±0.38
136	10.24±1.38	3.71±1.88	8.85 ±0.94	1.34±0.38	2.25±0.33
204	10.50±1.63	5.65±1.63	9.39 ±0.76	1.68±0.37	4.92±0.19
272	11.22±1.19	8.64±1.06	10.29±0.69	2.03±0.13	5.23±0.44
340	12.03±1.13	9.82±1.81	11.57±0.81	2.31±0.25	6.20±0.31

dimensional method. Thus, we will show some experimental results with different data sizes and scales.

We conduct experiments on three representative data sets, the image data with largest resolution, i.e., Ar, the data sets with largest size, i.e., Pie and Mnist. For illustration, we compare MRR with LDA, 2DLDA, 1DREG and GBR. Since KNN and SVM are evaluation methods and all the other approaches employ them for classification, we have not compared with them. For justice, these methods are all implemented in their original formulation, without using other accelerating strategies. With each fixed number of training points, we randomly select training points for 50 runs. With a naive MATLAB implementation, the calculations are made on a 3.2-GHz Windows machine. The computational time of different methods are listed in Tables VII and VIII and IX. The balance parameter α is also determined by cross validation.

There are also some observations from these results.

1) Among different methods on different data sets, GBR consumes the least time. Although MRR costs a little more time than GBR, it still consumes much less time than other one dimensional methods. Thus, it is suitable for dealing with large size matrix data. Besides, we can also see that two dimensional methods cost less time than its one dimensional counterparts.
2) Comparing the results on different image resolutions, we can see that dimensionality is the key factor in dominating the computational complexity. Certainly, with the increase of training points, all methods need more time. Nevertheless, compared with the influence of dimensionality, its effect is not so significant.

E. Classification With Different Parameters

In this section, we will provide some results with different α and k.

To show the influence of α, we employ KNN as the classifier and conduct experiments on three data sets, i.e., Umist, Coil and Usps. With the same setting as previous, we first determine α by cross validation and then vary this parameter near the determined value. With different numbers of training points, we report the mean accuracy of classification in Fig. 5.

As seen from Fig. 5, we can conclude that the parameter α takes influence on the final results. Nevertheless, its influence is not so significant if it changes within a certain range.

TABLE IX
COMPUTATIONAL TIME ON MNIST DATA WITH DIFFERENT NUMBERS OF TRAINING POINTS. THE DATA SCALE IS 28×28. (MEAN $\pm$ STD)

	LDA	2DLDA	1DREG	GBR	MRR
100	3.62±0.80	1.09±0.58	2.43 ±0.75	0.18±0.35	0.90±0.63
200	3.90±0.63	1.39±0.67	3.62 ±0.67	0.43±0.48	0.78±0.58
300	4.25±0.55	1.71±0.63	3.93 ±0.75	0.62±0.50	1.34±0.62
400	6.09±0.38	2.17±0.86	5.59 ±0.79	1.54±0.56	2.40±0.63
500	7.85±0.94	3.85±0.84	6.68 ±0.76	2.00±0.46	3.64±0.56

The second important parameter is the rank of regression, i.e., k. As stated above, this parameter can balance the influence of fitting error and the capacity for generalization. With different numbers of training points, we increase k from 1 to 4 and conduct experiments on another three data sets, i.e., Usps, Ar and Mnist. The results shown in Fig. 6 are also averaged for 50 independent runs.

As seen from the results in Fig. 6, we can see that with the increase of k, the fitting error decreases. Nevertheless, the classification accuracy does not always increase consistently. This is due to the fact that k is a parameter to balance the influences of fitting error and the capacity for generalization. These two factors are both important in dominating the performance of regression. GBR and 1DREG can be regarded as two extreme cases when $k = 1$ and $k = \min(m, n)$. These results validate the effectiveness in employing multiple rank regression vectors. Besides, when $k \geq 2$, experimental results show that the variance is limited. Thus, we determine $k = 2$ in all experiments.

VI. CONCLUSION

In this paper, we have proposed an efficient multiple rank regression model, i.e., MRR, for data matrix classification. Different from traditional regression approaches which reformulate the matrix data into a vector and use only one projecting vector, we have used several left projecting vectors and the same number of right projecting vectors to regress each matrix data to its label for each category. Essentially, MRR can be regarded as a tradeoff between the capacity of learning and generalization for regression. Plenty of experimental results have been proposed to show that MRR can not only achieve satisfied accuracy, but also has low computational complexity. Further research includes the extension of MRR to unsupervised and semi-supervised cases. We will also focus on the accelerating issue of our MRR algorithm.

APPENDIX

PROOF OF PROPOSITION 1

Proof: When $\mathbf{V}^{(r)}$ is fixed, the optimization problem in Eq. (10) is equivalent to the problem in Eq. (15). Considering that the optimization problem in Eq. (15) is convex to $\hat{\mathbf{U}}^{(r)}$ and b_r, we can derive its global optimization by setting its derivative to zero. In fact, the formulation in Eq. (15) is a regularized least regression problem. Thus, the results $\mathbf{U}^{(r)}$, derived from $\hat{\mathbf{U}}^{(r)}$, and b_r in Eq. (16) are the global solutions to the problem in Eq. (10). Similarly, we can conduct the same conclusion when $\mathbf{U}^{(r)}$ is fixed. ■

PROOF OF PROPOSITION 2

Proof: Since c classifiers are trained separately, we only need to prove that for each category, the objective function decreases. In other words, we only prove that the objective function in Eq. (10) will decrease in each iteration. Assume that we have derived $\mathbf{U}^{(r,s)}$, $\mathbf{V}^{(r,s)}$ and $b_r^{(s)}$ in the s-th iteration. In the $(s+1)$-th iteration, we fix $\mathbf{V}$ as $\mathbf{V}^{(r,s)}$ and optimize $\mathbf{U}^{(r)}$, b_r by minimizing the objective function in Eq. (11). Recalling the results in Proposition 1, we have the following inequality in Eq. (20)

$$\begin{aligned}
&\sum_{i=1}^{l}\left(\sum_{j=1}^{k}\left(\mathbf{u}_j^{(r,s+1)}\right)^T\mathbf{X}_i\mathbf{v}_j^{(r,s)}+b_r^{(s+1)}-y_{ir}\right)^2\\
&\quad+\alpha\sum_{j=1}^{k}Tr\left(\mathbf{u}_j^{(r,s+1)}\left(\mathbf{v}_j^{(r,s)}\right)^T\mathbf{v}_j^{(r,s)}\left(\mathbf{u}_j^{(r,s+1)}\right)^T\right)\\
&\leq\sum_{i=1}^{l}\left(\sum_{j=1}^{k}\left(\mathbf{u}_j^{(r,s)}\right)^T\mathbf{X}_i\mathbf{v}_j^{(r,s)}+b_r^{(s)}-y_{ir}\right)^2\\
&\quad+\alpha\sum_{j=1}^{k}Tr\left(\mathbf{u}_j^{(r,s)}\left(\mathbf{v}_j^{(r,s)}\right)^T\mathbf{v}_j^{(r,s)}\left(\mathbf{u}_j^{(r,s)}\right)^T\right). \qquad (20)
\end{aligned}$$

Similarly,

$$\begin{aligned}
&\sum_{i=1}^{l}\left(\sum_{j=1}^{k}\left(\mathbf{u}_j^{(r,s+1)}\right)^T\mathbf{X}_i\mathbf{v}_j^{(r,s+1)}+b_r^{(s+1)}-y_{ir}\right)^2\\
&\quad+\alpha\sum_{j=1}^{k}Tr\left(\mathbf{u}_j^{(r,s+1)}\left(\mathbf{v}_j^{(r,s+1)}\right)^T\mathbf{v}_j^{(r,s+1)}\left(\mathbf{u}_j^{(r,s+1)}\right)^T\right)\\
&\leq\sum_{i=1}^{l}\left(\sum_{j=1}^{k}\left(\mathbf{u}_j^{(r,s+1)}\right)^T\mathbf{X}_i\mathbf{v}_j^{(r,s)}+b_r^{(s+1)}-y_{ir}\right)^2\\
&\quad+\alpha\sum_{j=1}^{k}Tr\left(\mathbf{u}_j^{(r,s+1)}\left(\mathbf{v}_j^{(r,s)}\right)^T\mathbf{v}_j^{(r,s)}\left(\mathbf{u}_j^{(r,s+1)}\right)^T\right) \qquad (21)
\end{aligned}$$

Combining the results in Eq. (20) and (21), we will have the following inequality, which indicates the decrease of objective function during iteration

$$\mathcal{L}\left(\mathbf{U}^{(r,s+1)},\mathbf{V}^{(r,s+1)},b_r^{(s+1)}\right)\leq\mathcal{L}\left(\mathbf{U}^{(r,s)},\mathbf{V}^{(r,s)},b_r^{(s)}\right). \qquad (22)$$

■

PROOF OF PROPOSITION 3

Proof: Denote $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k]$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$, we have

$$\sum_{i}^{k}\mathbf{u}_i\mathbf{v}_i^T=\mathbf{U}\mathbf{V}^T \qquad (23)$$

On one hand, for any mn-dimensional vector $\mathbf{z}$, its matrix form is denoted as $\mathbf{Z} \in \mathbb{R}^{m\times n}$. Since $rank(\mathbf{Z}) \leq \min(m, n)$, there exits two matrixes, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$, such that $\mathbf{Z} = \mathbf{U}\mathbf{V}^T$. In other words, for any

mn-dimensional vector $\mathbf{z}$, we can find k couples of vectors, such that $\mathbf{Z} = \sum_i^k \mathbf{u}_i \mathbf{v}_i^T$, or equivalently, $\mathbf{z} = Vec(\sum_i^k \mathbf{u}_i \mathbf{v}_i^T)$.

On the other hand, for any $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ and $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$, $Vec(\sum_i^k \mathbf{u}_i \mathbf{v}_i^T) \in \mathbb{R}^{mn}$.

Combining the above two results, we get the conclusion. ■

REFERENCES

[1] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, and H. Zhang, "Multilinear discriminant analysis for face recognition," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 212–220, Jan. 2007.

[2] A. W.-K. Kong, D. D. Zhang, and M. S. Kamel, "A survey of palmprint recognition," *Pattern Recognit.*, vol. 42, no. 7, pp. 1408–1418, 2009.

[3] J. J. Koo, A. C. Evans, and W. J. Gross, "3-D brain MRI tissue classification on FPGAs," *IEEE Trans. Image Process.*, vol. 18, no. 12, pp. 2735–2746, Dec. 2009.

[4] G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice* (Neural Information Processing). Cambridge, MA: MIT Press, 2006.

[5] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ: Springer-Verlag, 2006.

[7] D. L. Donoho, "High-dimensional data analysis: The curses and blessings of dimensionality," in *Proc. Amer. Math. Soc. Conf. Math Challenges 21st Century*, 2000.

[8] D. Xu, S. Yan, S. Lin, T. S. Huang, and S.-F. Chang, "Enhancing bilinear subspace learning by element rearrangement," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 10, pp. 1913–1920, Oct. 2009.

[9] D. Cai, X. He, Y. Hu, J. Han, and T. Huang, "Learning a spatially smooth subspace for face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Mach. Learn.*, Mar. 2007, pp. 1–7.

[10] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 775–779, Jul. 1997.

[11] M. J. Lyons, J. Budynek, and S. Akamatsu, "Automatic classification of single facial images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 12, pp. 1357–1362, Dec. 1999.

[12] K. Ma and X. Tang, "Discrete wavelet face graph matching," in *Proc. Int. Conf. Image Process.*, 2001, pp. 217–220.

[13] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear subspace analysis of image ensembles," in *Proc. Comput. Vis. Pattern Recognit.*, 2003, pp. 93–99.

[14] I. Jolliffe, *Principal Component Analysis*, 2nd ed. New York: Springer-Verlag, 2002.

[15] J. Ye and Q. Li, "LDA/QR: An efficient and effective dimension reduction algorithm and its theoretical foundation," *Pattern Recognit.*, vol. 37, no. 4, pp. 851–854, 2004.

[16] X. He and P. Niyogi, "Locality preserving projections," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2003.

[17] F. Nie, S. Xiang, Y. Song, and C. Zhang, "Orthogonal locality minimizing globality maximizing projections for feature extraction," *Opt. Eng.*, vol. 48, no. 1, pp. 017202-1–017202-5, 2009.

[18] D. Cai, X. He, and J. Han, "Subspace learning based on tensor analysis," Dept. Comput. Sci., UIUC, Urbana, Tech. Rep. UIUCDCS-R-2005-2572, May 2005.

[19] J. Ye, R. Janardan, and Q. Li, "Two-dimensional linear discriminant analysis," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2004.

[20] M. Li and B. Yuan, "2D-LDA: A statistical linear discriminant analysis for image matrix," *Pattern Recognit. Lett.*, vol. 26, no. 5, pp. 527–532, 2005.

[21] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, and H. Zhang, "Multilinear discriminant analysis for face recognition," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 212–220, Jan. 2007.

[22] X. He, D. Cai, and P. Niyogi, "Tensor subspace analysis," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2005.

[23] F. Nie, S. Xiang, Y. Song, and C. Zhang, "Extracting the optimal dimensionality for local tensor discriminant analysis," *Pattern Recognit.*, vol. 42, no. 1, pp. 105–114, 2009.

[24] K. R. Gabriel, "Generalised bilinear regression," *Biometrika*, vol. 85, no. 3, pp. 689–700, 1998.

[25] P. Zhang and J. Peng, "Efficient regularized least squares classification," in *Proc. Comput. Vis. Pattern Recognit. Workshop*, 2004, pp. 1–98.

[26] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. New York: Chapman & Hall, 1993.

[27] D. Cai, X. He, and J. Han, "SRDA: An efficient algorithm for large-scale discriminant analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 1–12, Jan. 2008.

[28] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, 1995, pp. 1137–1143.

Chenping Hou (M'12) received the B.S. and Ph.D. degrees in applied mathematics from the National University of Defense Technology, Changsha, China, in 2004 and 2009, respectively.

He is currently a Lecturer with the Department of Mathematics and Systems Science, National University of Defense Technology. His current research interests include pattern recognition, machine learning, data mining, and computer vision.

Dr. Hou is a member of the Association for Computing Machinery.

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

He is currently a Research Assistant Professor with the University of Texas, Arlington. His current research interests include machine learning and its application fields, such as pattern recognition, data mining, computer vision, image processing, and information retrieval.

Dongyun Yi received the B.S. degree from Nankai University, Tianjin, China, and the M.S. and Ph.D. degrees from the National University of Defense Technology, Changsha, China.

He was a Visiting Researcher with the University of Warwick, Coventry, U.K., in 2008. He is a Professor with the Department of Mathematics and Systems Science, National University of Defense Technology. His current research interests include statistics, systems science, and data mining.

Yi Wu received the B.S. and M.S. degrees in applied mathematics from the National University of Defense Technology, Changsha, China, in 1981 and 1988, respectively.

He is currently a Professor with the Department of Mathematics and Systems Science, National University of Defense Technology. He was a Visiting Researcher with New York State University, New York, in 1999. His current research interests include applied mathematics, statistics, and data mining.