Pattern Recognition 47 (2014) 454–469

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Multiple rank multi-linear SVM for matrix data classification

Chenping Hou [a,*], Feiping Nie [b], Changshui Zhang [c], Dongyun Yi [a], Yi Wu [a]

[a] Department of Mathematics and Systems Science, National University of Defense Technology, Changsha 410073, China
[b] Department of Computer Science and Engineering, University of Texas, Arlington 76019, USA
[c] Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 23 August 2012
Received in revised form
5 June 2013
Accepted 4 July 2013
Available online 15 July 2013

Keywords:
Pattern recognition
Matrix data classification
Learning capacity
Generalization
SVM
STM

ABSTRACT

Matrices, or more generally, multi-way arrays (tensors) are common forms of data that are encountered in a wide range of real applications. How to classify this kind of data is an important research topic for both pattern recognition and machine learning. In this paper, by analyzing the relationship between two famous traditional classification approaches, i.e., SVM and STM, a novel tensor-based method, i.e., multiple rank multi-linear SVM (MRMLSVM), is proposed. Different from traditional vector-based and tensor based methods, multiple-rank left and right projecting vectors are employed to construct decision boundary and establish margin function. We reveal that the rank of transformation can be regarded as a tradeoff parameter to balance the capacity of learning and generalization in essence. We also proposed an effective approach to solve the proposed non-convex optimization problem. The convergence behavior, initialization, computational complexity and parameter determination problems are analyzed. Compared with vector-based classification methods, MRMLSVM achieves higher accuracy and has lower computational complexity. Compared with traditional supervised tensor-based methods, MRMLSVM performs better for matrix data classification. Promising experimental results on various kinds of data sets are provided to show the effectiveness of our method.

1. Introduction

Matrices, or more generally, multi-way arrays (tensors) are common forms of data that are encountered in a wide range of real applications. For example, all raster images are essentially digital readings of a grid of sensors and matrix analysis is widely applied in image processing, e.g., photorealistic images of faces [1] and palms [2], and medical images [3]. In web search, one can easily get a large volume of images represented in the form of matrix. Besides, in video data mining, the data at each time frame is also a matrix. Therefore, matrix data analysis, in particular, classification, has become one of the most important topics for both pattern recognition and computer vision.

Classification is arguably the most often task in pattern recognition and relevant techniques are abundant in the literature. Standard methods, such as *K*-nearest neighborhoods classifier (KNN) [4], support vector machine (SVM) [5,6] and one-dimensional regression methods [7] are widely used in many fields. Among these approaches, some of them are similarity based, such as KNN, and some of them are margin based, such as SVM. Due to its practical effectiveness and theoretical soundness, SVM and its variants, especially linear SVM, have been widely used in many real applications [8,9]. For example, SVM has been combined with factorization machine (FM) [10] for spammer discovering in social networks [11]. Nevertheless, traditional classification methods are usually vector-based. They assume that the inputs of an algorithm are vectors, not matrix data or tensor data. When they are applied to matrix data, the matrix structure must be collapsed to make vector inputs for the algorithm. One common way is connecting each row (or column) of a matrix to reformulate a vector.

Although traditional classification approaches have achieved satisfactory performance in many cases, they may lack efficiency in managing matrix data by simply reformulating them into vectors. The main reasons are as follows: (1) When we reformulate a matrix as a vector, the dimensionality of this vector is often very high. For example, for a small image of resolution 256×256, the dimensionality of reformulated vector is 65,536. The performances of traditional vector based methods will degrade due to the increase of dimensionality [12,13]. (2) With the increase of dimensionality, the computation time will increase drastically. For example, the computational complexity of SVM is closely related to the dimensionality of input data [9]. If the matrix scale is large, traditional approaches cannot be implemented in this scenario. (3) When a matrix is collapsed as a vector, the spatial correlations of the matrix will be lost [14]. For example, if an image of $m \times n$ is represented as a mn-dimensional vector, it suggests that the image is specified by mn independent variables.

* Corresponding author. Tel.: +86 731 8457 3238.
E-mail address: hcpnudt@hotmail.com (C. Hou).

0031-3203/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2013.07.002

However, in practice, there are generally only a few interested aspects about an image, and the degree of freedom of the model is far less than mn.

In order to solve the above mentioned problems, a lot of researchers have proposed many approaches. There are mainly two types of methods. The first type of methods reduces dimensionality of original tensor data and then employs some off-the-shelf classifiers on the projected vector data. For example, in image processing field, there have been many feature based methods, such as the elastic graph model which can preserve spatial information in a compact dimensionality [15]. Recently, a lot of interests have been conducted on tensor-based approaches for matrix data analysis. Vasilescu et al. have proposed the famous tensor face for face recognition [16]. After that, a lot of researchers have also extended traditional subspace learning methods, such as principal component analysis (PCA) [17], linear discriminant analysis (LDA) [18], locality preserving projection (LPP) [19], etc., into their tensor counterparts [1,20–24]. Nevertheless, since most of these methods are unsupervised, they have lost label information in learning subspace. They cannot be used for tensor data classification directly.

The second type of methods can classify tensor data directly [25–30]. For example, Tao et al. have proposed a framework to extend traditional approaches into their tensor counterparts, e.g., support tensor machine (STM) [25,26]. Other related works in multiple view learning have also used the same strategy to incorporate data representations from different views [31]. When we apply these tensor-based approaches to classify matrix data, their performances can also be improved since their training error is often large. For example, STM only uses one left projecting vector together with one right projecting vector. Its training error is larger than the following proposed approach.

In this paper, by discovering the relationship between SVM and STM, we introduce a novel matrix classification model using multiple rank projections. It is named as multiple rank multilinear SVM (MRMLSVM). Instead of converting an $m \times n$ matrix into an mn-dimensional vector as in traditional linear SVM, and using only one left and one right transformation vector as in STM, we employ two groups of transformation matrices in designing constraints and establishing objective function. More importantly, there are several transformation vectors in each group. Essentially, we discover that the number of transformation vectors is a tradeoff parameter for the capacity of learning and generalization of a learning machine. Besides, we have also proposed an effective optimization strategy and some deep analyses, including convergence analysis, initialization, computational complexity and parameter determination. Compared with other vector based methods, such as LDASVM (LDA for subspace learning and linear SVM for classification) and linear SVM, and tensor-based approaches, such as 2DLDASVM (2DLDA for subspace learning and linear SVM for classification) and STM, it achieves more promising results in matrix data classification. Compared with traditional classification approaches in converting matrix into vector, the computational complexity is also reduced. Plenty of experiments on different kinds of data are presented for illustration.

It is worthwhile to highlight the contribution of our algorithm:

(1) We have revealed the relationship between traditional linear SVM and STM. Based on this analysis, we have proposed a novel approach, i.e., MRMLSVM, for matrix data classification. Compared with other related vector based and tensor based approaches, it can achieve promising classification accuracy.
(2) We have provided an effective way to solve the proposed non-convex problem. Compared with other vector based counterparts, its computational complexity is low, especially when the data scale is large. Experimental results have been proposed for demonstration.
(3) The most important parameter in MRMLSVM is the rank of regression. We have revealed its essence. It can be regarded as a parameter to balance the capacity of learning and generalization for a learning machine. This is important for tightening the relationship between two famous learning methods, i.e., SVM and STM.
(4) MRMLSVM is just an instance in using multiple rank projections. Intrinsically, we can regard it as a common model. Other linear methods can also be extended by the similar way.

The rest of this paper is organized as follows. In Section 2, we will provide some notations and some related works. In Section 3, by analyzing the relation between linear SVM and STM, we propose the MRMLSVM algorithm in details, together with an effective way in solving this problem. We present the performance analyses, including convergence behavior, initialization, computational complexity and parameter determination in Section 4. Section 5 provides some promising comparing results on various kinds of data sets, followed by the conclusions and future works in Section 6.

2. Notations and related works

In this section, we would like to introduce some representative works, including 2DLDA, linear SVM and STM, since 2DLDA is a typical supervised two-dimensional dimensionality reduction method and our algorithm has close relationship with linear SVM and STM. Before going into the details, let us introduce some notations at first.

2.1. Notations

In this paper, we try to solve a supervised matrix data classification problem. Besides, we only introduce our algorithm for matrix data (two order tensor) in binary classification task. As we will explain later, it is direct to extend our approach to any orders of tensor and any numbers of categories.

Denote $\{\mathbf{X}_i \in \mathbb{R}^{m\times n} | i = 1, 2, \ldots, l\}$ as a set of training examples. The associated class label vectors are $\{y_1, y_2, \ldots, y_l\}$, where $y_i = 1$ iff $\mathbf{X}_i$ belongs to the first category and $y_i = -1$ otherwise. m and n are the first and second dimensions of each matrix data respectively. l is the number of training points. Additionally, we also have t testing points $\{\mathbf{X}_i \in \mathbb{R}^{m\times n} | i = l+1, l+2, \ldots, l+t\}$. The vectorization of $\mathbf{X}_i$, denoted as $\mathbf{x}_i$, is formulated by connecting each column vector of $\mathbf{X}_i$ for $i = 1, 2, \ldots, l+t$.

Define $\mathbf{u}_j \in \mathbb{R}^m$ and $\mathbf{v}_j \in \mathbb{R}^n$ as the j-th left and right projection vectors, where $j = 1, 2, \ldots, k$ and k is rank of linear transformation. Other important notations are summarized in Table 1. We will explain their concrete meanings when they are firstly used.

Table 1
Notations.

l	Number of training points
t	Number of testing points
m	The first dimensionality of matrix data
n	The second dimensionality of matrix data
k	Rank of linear transformation
$\mathbf{X}_i \in \mathbb{R}^{m\times n}$	The i-th training matrix data
$\mathbf{x}_i \in \mathbb{R}^{mn}$	The vectorization of $\mathbf{X}_i$
$y_i \in \{1, -1\}$	The label vector of $\mathbf{X}_i$
$\mathbf{u}_i \in \mathbb{R}^m$	The i-th left regression vector
$\mathbf{v}_i \in \mathbb{R}^n$	The i-th right regression vector
$\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_k]$	The left regression matrix
$\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_k]$	The right regression matrix
$\|\cdot\|$	The Frobenius norm
$\otimes$	The Kronecker product

2.2. 2DLDA

2DLDA is one of the most important supervised two-dimensional subspace learning methods. It can be regarded as the two-dimensional extension of traditional LDA approach. Denote $\mathcal{M}_i$ as the set of training points in the i-th class, where $\mathcal{M}_i$ has l_i samples. Let $\overline{\mathbf{X}}_i=(1/l_i)\sum_{\mathbf{X}_i\in\mathcal{M}_i}\mathbf{X}_i$ be the mean of samples in the i-th class for $1\le i\le c$. Denote $\overline{\mathbf{X}}=(1/l)\sum\mathbf{X}_i$ as the mean of all training data.

2DLDA tries to find two transformation matrices $\mathbf{L}$ and $\mathbf{R}$, which project $\mathbf{X}_i$ to its low-dimensional embedding, i.e., $\mathbf{Z}_i$, by $\mathbf{Z}_i=\mathbf{L}^T\mathbf{X}_i\mathbf{R}$. Define the within-class distances D_w and between-class distance D_b as follows:

$$D_w=\sum_{j=1}^{c}\sum_{\mathbf{X}_i\in\mathcal{M}_j}\|\mathbf{X}_i-\overline{\mathbf{X}}_j\|^2,$$
$$D_b=\sum_{j=1}^{c}l_j\|\overline{\mathbf{X}}_j-\overline{\mathbf{X}}\|^2. \tag{1}$$

where $\|\cdot\|$ is the Frobenius norm of a matrix. Intuitively, D_b measures the sum of divergence between any two classes and D_w is the sum of data variance for each category.

Similar to LDA, in the low-dimensional space, the optimal transformation matrices $\mathbf{L}$ and $\mathbf{R}$ in 2DLDA should minimize $\tilde{D}_w$ and maximize $\tilde{D}_b$, the low-dimensional counterparts of D_w and D_b shown as follows:

$$\tilde{D}_w=\mathrm{Tr}\left(\sum_{j=1}^{c}\sum_{\mathbf{X}_i\in\mathcal{M}_j}\mathbf{L}^T(\mathbf{X}_i-\overline{\mathbf{X}}_j)\mathbf{R}\mathbf{R}^T(\mathbf{X}_i-\overline{\mathbf{X}}_j)^T\mathbf{L}\right),$$
$$\tilde{D}_b=\mathrm{Tr}\left(\sum_{j=1}^{c}l_j\mathbf{L}^T(\overline{\mathbf{X}}_j-\overline{\mathbf{X}})\mathbf{R}\mathbf{R}^T(\overline{\mathbf{X}}_j-\overline{\mathbf{X}})^T\mathbf{L}\right). \tag{2}$$

Since it is difficult to derive the optimal $\mathbf{L}$ and $\mathbf{R}$ simultaneously, 2DLDA solves the above problem in Eq. (2) 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 [21].

As seen from above formulation, although 2DLDA inherits the discriminative power in deriving low-dimensional representations, it cannot be used for classification directly.

2.3. Linear SVM

SVM is one of the most popular classifier in real applications. It maximizes the margin for two categories. More concretely, denote $\mathbf{w}$ as the vector orthogonal to the decision boundary and b as a scalar "offset" term, we can write the decision boundary as $\mathbf{w}^T\mathbf{x}+b=0$ for any vector data $\mathbf{x}$. To relaxed the hard constraints, one common way is to soften it as $y_i(\mathbf{w}^T\mathbf{x}_i+b)\ge 1-\xi_i$, where $\xi_i\ge 0$ is the slack variable. Since the margin is proportionate to $1/\|\mathbf{w}\|$, the concrete formulation of linear SVM can be defined as follows:

$$\begin{aligned}\arg\min_{\mathbf{w},b,\boldsymbol{\xi}}\quad & \frac{1}{2}\mathbf{w}^T\mathbf{w}+C\sum_{i=1}^{l}\xi_i\\ \text{s.t.}\quad & y_i(\mathbf{w}^T\mathbf{x}_i+b)\ge 1-\xi_i\\ & \xi_i\ge 0\quad \text{for } i=1,2,\ldots,l,\end{aligned} \tag{3}$$

where $C>0$ is the regularization parameter and $\boldsymbol{\xi}=[\xi_1,\xi_2,\ldots,\xi_l]^T$ consists of all slack variables.

Let us explain the meaning of each function. The objective function aims to maximize the margin and the constraints indicate that the training points should be correctly classified by the relaxed decision function $\mathbf{w}^T\mathbf{x}+b$. The optimization problem in Eq. (3) can be solved by casting it to its dual form. We can also use the kernel trick to manipulate nonlinear classification tasks. See more details in [32].

As seen from the formulation in Eq. (3), traditional linear SVM only involves vector data. In manipulating matrix data, we can only employ its vectorization as the input. This would lose the spatial information [14] of original matrix data and enlarge the computational cost.

2.4. STM

STM is the tensor extension of traditional SVM [25]. When it is used to manipulate matrix data, it uses two transformation vectors $\mathbf{u}$ and $\mathbf{v}$ to replace the original transformation vector $\mathbf{w}$ in Eq. (3). More concretely, the margin function is replaced by $\|\mathbf{u}\otimes\mathbf{v}\|^2$ and the transformation $\mathbf{w}^T\mathbf{x}_i$ is replaced by $\mathbf{u}^T\mathbf{X}_i\mathbf{v}$. STM has the following formulation:

$$\begin{aligned}\arg\min_{\mathbf{u},\mathbf{v},\xi,b}\quad & \frac{1}{2}\|\mathbf{u}\otimes\mathbf{v}\|^2+C\sum_{i=1}^{l}\xi_i\\ \text{s.t.}\quad & y_i(\mathbf{u}^T\mathbf{X}_i\mathbf{v}+b)\ge 1-\xi_i\\ & \xi_i\ge 0\quad \text{for } i=1,2,\ldots,l,\end{aligned} \tag{4}$$

where $\otimes$ indicates the Kronecker product between two vectors and $\mathbf{u}\otimes\mathbf{v}=\mathrm{Vec}(\mathbf{u}\mathbf{v}^T)$.

As stated in [25], since the derivation of $\mathbf{u}$ is related to $\mathbf{v}$, we cannot solve the optimization problem in Eq. (4) directly. One possible way is optimizing them alternatively. Moreover, when $\mathbf{v}$ is fixed, the problem in Eq. (4) amounts to a standard SVM problem. Similarly, when $\mathbf{u}$ is fixed, we can use the same way to derive $\mathbf{v}$. See more details in [25].

3. Multiple rank multi-linear SVM

In this section, we would like to introduce our multiple rank multi-linear SVM (MRMLSVM) algorithm. Before going into the details, we analyze the relationship between SVM and STM. Then, the formulation of MRMLSVM is introduced step by step. Since the formulated problem is not convex, we propose an effective method to find the approximated solution in an alternative way. To show the effectiveness theoretically, we will also provide some deep analyses in the next section.

3.1. Relationship between SVM and STM

Comparing the formulation of SVM in Eq. (3) with that of STM in Eq. (4), we know that the transformation vector $\mathbf{w}$ in SVM's objective function has been replaced by $\mathbf{u}\otimes\mathbf{v}$ as that in STM. Correspondingly, the constraints have been changed from $y_i(\mathbf{w}^T\mathbf{x}_i+b)\ge 1-\xi_i$ to $y_i(\mathbf{u}^T\mathbf{X}_i\mathbf{v}+b)\ge 1-\xi_i$.

Notice that

$$\begin{aligned}\mathbf{u}^T\mathbf{X}_i\mathbf{v}&=\mathrm{Tr}(\mathbf{u}^T\mathbf{X}_i\mathbf{v})=\mathrm{Tr}(\mathbf{X}_i\mathbf{v}\mathbf{u}^T)=\mathrm{Tr}(\mathbf{X}_i(\mathbf{v}\mathbf{u}^T))\\ &=\mathrm{Tr}((\mathbf{u}\mathbf{v}^T)^T\mathbf{X}_i)=(\mathrm{Vec}(\mathbf{u}\mathbf{v}^T))^T\mathrm{Vec}(\mathbf{X}_i)\end{aligned} \tag{5}$$

and

$$\|\mathbf{u}\otimes\mathbf{v}\|^2=\|\mathrm{Vec}(\mathbf{u}\mathbf{v}^T)\|^2. \tag{6}$$

Here $\mathrm{Vec}(\cdot)$ is the vectorization of a matrix (by connecting each column of this matrix).

As seen from above deduction, when we employ STM to classify matrix data, it is equivalent to the employment of SVM on the vectorization of corresponding data, provided that the constraint is $\mathbf{w}=\mathrm{Vec}(\mathbf{u}\mathbf{v}^T)$. More concretely, we have the following discussions.

On one hand, STM can be regarded as a special case of SVM by adding the constraint that $\mathbf{w}=\mathrm{Vec}(\mathbf{u}\mathbf{v}^T)$. It indicates that the corresponding $\mathbf{w}$ is determined by only $m+n$ variables. In many real applications, such as image processing, a real matrix of size $m\times n$ has mn elements. It cannot be simply modeled by just $m+n$ independent variables [14]. In other words, this added constraint is too strict since we only have $m+n$ free variables to model $m\times n$-dimensional vectors derived from original matrices. It will make the model lack of flexibility in modeling matrix data.

By contrast, $\mathbf{w}$ in SVM has mn free variables. From the view of optimization, the feasible region of STM is much smaller than that of SVM due to this additional constraints. Consequently, compared with SVM who can find the global solution in a larger feasible region, the objective function value of STM is larger, which indicates that it has larger training error.

On the other hand, compared with STM, SVM has larger degree of freedom in selecting feasible $\mathbf{w}$ since it regards mn elements in the matrix independently. Nevertheless, it treats traditional vector data and vectorized matrix data equally. Compared with traditional vector data, the matrix data also have some spatial dependence. For example, all pixels of an image cannot be treated independently [14]. SVM neglects the spatial dependence of a matrix data and treats them independently. In other words, this model has too many free variables and it will cause the problem of over fitting.

Besides, the above connection between SVM and STM also holds when the order of tensor is larger than 2. For illustration, we take three order tensor as an example. Extension to any order can be derived in a similar way. Assume $\mathbf{X}\in\mathbb{R}^{m_1\times m_2\times m_3}$ is a three order tensor. The corresponding transformation vectors are $\mathbf{u}^{(j)}\in\mathbb{R}^{m_j}$ for $j=1,2,3$. To show their relationship, we only need to prove that Eqs. (5) and (6) also hold in this scenario.

Notice that

$$\|\prod_{j=1}^{3}\otimes\mathbf{u}^{(j)}\|^2=\sum_{p_1=1}^{m_1}\sum_{p_2=1}^{m_2}\sum_{p_3=1}^{m_3}(u_{p_1}^{(1)}u_{p_2}^{(2)}u_{p_3}^{(3)})^2=\|\mathrm{Vec}\left(\prod_{j=1}^{3}\otimes\mathbf{u}^{(j)}\right)\|^2$$

and

$$\mathbf{X}\times_1\mathbf{u}^{(1)}\times_2\mathbf{u}^{(2)}\times_3\mathbf{u}^{(3)}=\sum_{p_1=1}^{m_1}\sum_{p_2=1}^{m_2}\sum_{p_3=1}^{m_3}X_{p_1,p_2,p_3}u_{p_1}^{(1)}u_{p_2}^{(2)}u_{p_3}^{(3)}=\left(\mathrm{Vec}\left(\prod_{j=1}^{3}\otimes\mathbf{u}^{(j)}\right)\right)^T\mathrm{Vec}(\mathbf{X})$$

where $\times_1$ is the model-1 time between a tensor and a vector and similar to others. $\mathrm{Vec}(\cdot)$ is the vectorization of a three order tensor by first connecting column of each splice and then connecting formulated vectors from each splice. See more details in [25].

As seen from above two equations, we know that STM can be regarded as special case of SVM, provided that $\mathbf{w}=\mathrm{Vec}(\prod_{j=1}^{3}\otimes\mathbf{u}^{(j)})$. It also holds in the matrix scenario since $\mathrm{Vec}(\mathbf{u}\mathbf{v}^T)=\mathrm{Vec}(\mathbf{u}\otimes\mathbf{v})$. Additionally, the above deductions can also be conducted on higher order tensor.

3.2. Multiple rank multi-linear constraints

Based on above analysis, we know that the constraints of STM and SVM have their own merits and disadvantages. To relax the too strict constraints in STM and avoid over fitting in SVM, instead of using merely one couple of projecting vectors, i.e., the left projecting vector $\mathbf{u}$ and right projecting vector $\mathbf{v}$, we propose to use k couples of left projecting vectors and right projecting vectors in our MRMLSVM formulation. They are denoted as $\{\mathbf{u}_j\}_{j=1}^{k}$ and $\{\mathbf{v}_j\}_{j=1}^{k}$. More concretely, the constraints in Eq. (4) becomes

$$\begin{aligned}&y_i(\mathbf{u}_1^T\mathbf{X}_i\mathbf{v}_1+\mathbf{u}_2^T\mathbf{X}_i\mathbf{v}_2+\cdots+\mathbf{u}_k^T\mathbf{X}_i\mathbf{v}_k+b)\geq 1-\xi_i\\&\xi_i\geq 0\quad\text{for } i=1,2,\ldots,l.\end{aligned}\tag{7}$$

Denote $\mathbf{U}=[\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_k]\in\mathbb{R}^{m\times k}$ and $\mathbf{V}=[\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k]\in\mathbb{R}^{n\times k}$, the constraints in Eq. (7) can be reformulated as follows:

$$\begin{aligned}&y_i(\mathrm{Tr}(\mathbf{U}^T\mathbf{X}_i\mathbf{V})+b)\geq 1-\xi_i\\&\xi_i\geq 0\quad\text{for } i=1,2,\ldots,l.\end{aligned}\tag{8}$$

Intuitively, compared with the employment of only one couple of projecting vectors, there are totally $k(m+n)$ free variables in our formulations. Since $k\geq 2$, it is more than that of STM who only has $m+n$ free variables. In other words, the too strict constraint $\mathbf{w}=\mathrm{Vec}(\mathbf{u}\mathbf{v}^T)$ in STM has been relaxed by inducing more free parameters. This constraint is not required to follow and our model is more flexible in characterizing matrix data. Besides, one couple of projecting vectors is a special case of our setting when $\mathbf{u}_j=\mathbf{0}$, $\mathbf{v}_j=\mathbf{0}$ for $j\geq 2$. More importantly, we have the following proposition which can reveal the essence of parameter k, the rank of projecting vectors.

Proposition 1. *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 $\mathrm{Vec}(\sum_i^k\mathbf{u}_i\mathbf{v}_i^T)$ is mn. Here $\mathrm{Vec}(\cdot)$ represents the vectorization of a matrix.*

The proof is listed in Appendix. Based on above proposition, we have the following discussions:

(1) When $k=\min(m,n)$ and the constraints are $\mathrm{Tr}((\mathbf{U}^{(r)})^T\mathbf{X}_i\mathbf{V}^{(r)})\geq 1-\xi_i$, the above proposition indicates that the searching space for $\mathbf{U}^{(r)}(\mathbf{V}^{(r)})^T$ is very similar to the feasible region determined by $(\mathrm{Vec}(\sum_i^k\mathbf{u}_i(\mathbf{v}_i)^T))^T\mathrm{Vec}(\mathbf{X}_i)$. In other words, $(\mathrm{Vec}(\sum_i^k\mathbf{u}_i(\mathbf{v}_i)^T))^T\mathrm{Vec}(\mathbf{X}_i)$ and $\mathrm{Tr}(\mathbf{U}^{(r)}\mathbf{X}_i\mathbf{V}^{(r)})$ are more likely to be the same. That is to say, the constraints in Eq. (8) are close to the corresponding constraint of SVM.

(2) As shown in above proposition, SVM has larger k and more free parameters ($m\times n$). Thus, the feasible region of SVM is larger and it can find the global optimization. Consequently, the training error is often smaller. Nevertheless, since it has too many parameters and the model is too complicated, it tends to be over fitting. When we use smaller k as in STM ($k=1$). The over fitting problem is avoided. Nevertheless, it will add too strict constraint since we only have $m+n$ free parameters. Compared with SVM who can find the global optimization in a larger feasible region, the training error of STM is often larger.

3.3. Formulation

Motivated by the above comparison, we now introduce our algorithm formally. Mathematically, MRMLSVM has the constraints shown in Eq. (8). Similar to SVM and STM, the objective function of MRMLSVM should describe the margin. Considering the objective functions of SVM and STM shown in Eqs. (3) and (4), we have

$$\begin{aligned}\sum_{j=1}^{k}\mathbf{u}_j^T\mathbf{X}_i\mathbf{v}_j&=\sum_{j=1}^{k}\mathrm{Tr}(\mathbf{u}_j^T\mathbf{X}_i\mathbf{v}_j)\\&=\sum_{j=1}^{k}\mathrm{Tr}(\mathbf{X}_i\mathbf{v}_j\mathbf{u}_j^T)=\mathrm{Tr}(\mathbf{X}_i(\sum_{j=1}^{k}\mathbf{v}_j\mathbf{u}_j^T))\\&=\mathrm{Tr}(\mathbf{X}_i\mathbf{V}\mathbf{U}^T)=\mathrm{Tr}((\mathbf{U}\mathbf{V}^T)^T\mathbf{X}_i)=(\mathrm{Vec}(\mathbf{U}\mathbf{V}^T))^T\mathrm{Vec}(\mathbf{X}_i)\end{aligned}\tag{9}$$

Comparing the formulated constraint of MRMLSVM (in Eq. (8)) with that of SVM (in Eq. (3)), we know that the tensor counterpart of $\mathbf{w}$ in Eq. (3) should be $\mathbf{U}\mathbf{V}^T$. Since the objective function of SVM is

$$\frac{1}{2}\mathrm{Tr}(\mathbf{w}\mathbf{w}^T)+C\sum_{i=1}^{l}\xi_i.$$

Then, the tensor extension of original objective function is

$$\frac{1}{2}\mathrm{Tr}\left(\mathbf{U}\mathbf{V}^T\mathbf{V}\mathbf{U}^T\right)+C\sum_{i=1}^{l}\xi_i.\tag{10}$$

In summary, by combing the constraints in Eq. (8) with the objective function in Eq. (10), we formulate our MRMLSVM

algorithm as follows:

$$\begin{aligned}\arg\min_{\mathbf{U},\mathbf{V},\xi,b}\quad & \frac{1}{2}\mathrm{Tr}\left(\mathbf{U}\mathbf{V}^T\mathbf{V}\mathbf{U}^T\right)+C\sum_{i=1}^{l}\xi_i\\ \text{s.t.}\quad & y_i(\mathrm{Tr}(\mathbf{U}^T\mathbf{X}_i\mathbf{V})+b)\geq 1-\xi_i\\ & \xi_i\geq 0\quad \text{for } i=1,2,\ldots,l,\end{aligned}\tag{11}$$

where $\mathbf{U}=[\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_k]$ and $\mathbf{V}=[\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k]$ are defined as previous.

As seen from above formulation of MRMLSVM, we notice that there is one important parameter k. We would like to reveal its essence.

First, we would like to show the influence of k to training error. As seen from above formulation and following solving strategy, we assume $1<k<\min(m,n)$ and use a similar way in formulating and solving the problem as in STM. More importantly, if we use the solution of STM to initialize MRMLSVM, we have the following proposition.

Proposition 2. *If we use the optimal value* $\mathbf{v}^*$ *of STM to initialize MRMLSVM by setting* $\mathbf{v}_1=\mathbf{v}^*$ *and other* $\mathbf{v}$ *by any values satisfied* $[\mathbf{v}_2,\mathbf{v}_3,\ldots,\mathbf{v}_k]\neq\mathbf{0}$. *The optimal function value of MRMLSVM is no larger than that of STM.*

The proof is listed in Appendix. As stated in above proposition, if we use this kind initialization, the training error of our method is no larger than that of STM. Besides, experimental results in Section 5.2 also show that other kinds of initializations (mentioned in Section 4.2) have similar training error, which is also smaller than that of STM. On the contrary, since SVM has largest k and it can find the global optimization, its feasible region is larger than that of MRMLSVM and its training error is no larger than that of MRMLSVM. Experimental results in Section 5.2 also validate this analysis. In summary, if we use STM to initialize MRMLSVM, the larger k indicates the smaller training error.

Second, we would like to show the influence of k to the extent in avoiding over fitting. As what we have mentioned in Proposition 1, SVM has the largest k and most free parameters ($m\times n$). It trends to be over fitting, when we use smaller k. The over fitting problem can be solved. In summary, the larger k indicates the more likely over fitting occurs.

Finally, based on above two points, we know that k can be regarded as a parameter to balance the training error and the extent in avoiding over fitting. In learning theory, training error can measure the capacity of learning and the extent in avoiding over fitting can measure the capacity of generalization [5]. Therefore, k is a parameter to balance these two capacities in essence. Recalling the basic rule of learning theory, we know that these two capacities cannot be improved simultaneously [5]. In our following formulations, we assume $1<k<\min(m,n)$. Thus, our method is a general tradeoff between two famous methods, i.e., SVM and STM.

3.4. Solution

In this section, since the problem in Eq. (11) is not jointly convex with respect to $\mathbf{U}$ and $\mathbf{V}$, we will try to find an approximated solution to the proposed problem. Recalling the basic solving procedure of STM, we would like to optimize $\mathbf{U}$ and $\mathbf{V}$ in an alternative way. More concretely, we fix one parameter and optimize the other one. Before going into details, we would like to introduce a lemma at first.

Lemma 1. *Assume that* $\mathbf{A}$, $\mathbf{B}$ *and* $\mathbf{C}$ *are three matrices and* $\mathbf{ABC}$ *exists, we have*

(1) $\mathrm{Vec}(\mathbf{ABC})=(\mathbf{C}^T\otimes\mathbf{A})\mathrm{Vec}(\mathbf{B})$.
(2) $\mathrm{Tr}(\mathbf{A}^T\mathbf{B})=\mathrm{Vec}(\mathbf{A})^T\mathrm{Vec}(\mathbf{B})$.

The proof of this lemma is direct and we would like to omit it.

Based on Lemma 1, we have the following proposition, which is vital for our following solution.

Corollary 1.

$$\mathrm{Tr}(\mathbf{U}\mathbf{V}^T\mathbf{V}\mathbf{U}^T)=\mathrm{Tr}(\mathbf{U}^T\mathbf{U}\mathbf{V}^T\mathbf{V})=(\mathrm{Vec}(\mathbf{U}))^T((\mathbf{V}^T\mathbf{V})\otimes\mathbf{I}_{m\times m})\mathrm{Vec}(\mathbf{U})\tag{12}$$

where $\mathbf{I}_{m\times m}$ *represents the* $m\times m$ *identity matrix.*

The proof is listed in Appendix. See more details there.

(1) *Fixing* $\mathbf{V}$ *and optimizing* $\mathbf{U}$ *and* b. As seen from Eq. (11), we reformulate the optimization problem of MRMLSVM as follows:

$$\begin{aligned}\arg\min_{\mathbf{U},\mathbf{V},\xi,b}\quad & \frac{1}{2}\mathrm{Tr}\left(\mathbf{U}\mathbf{V}^T\mathbf{V}\mathbf{U}^T\right)+C\sum_{i=1}^{l}\xi_i\\ \text{s.t.}\quad & y_i\left(\sum_{j=1}^{k}\mathbf{u}_j^T\mathbf{X}_i\mathbf{v}_j+b\right)\geq 1-\xi_i\\ & \xi_i\geq 0\quad \text{for } i=1,2,\ldots,l.\end{aligned}\tag{13}$$

When we fix $\mathbf{V}$, or equivalently, $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k$, we need to compute a set of $\mathbf{u}_j$ for $j=1,2,\ldots,k$ and b. First, let us reformulate the problem in Eq. (13) as follows. Denote

$$\hat{\mathbf{f}}_i=\begin{bmatrix}\mathbf{X}_i\mathbf{v}_1\\ \mathbf{X}_i\mathbf{v}_2\\ \vdots\\ \mathbf{X}_i\mathbf{v}_k\end{bmatrix}_{mk\times 1},\quad \hat{\mathbf{u}}=\begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\\ \vdots\\ \mathbf{u}_k\end{bmatrix}_{mk\times 1},\mathbf{D}=(\mathbf{V}^T\mathbf{V})\otimes\mathbf{I}_{m\times m}\tag{14}$$

Then, recall that $\hat{\mathbf{u}}=\mathrm{Vec}(\mathbf{U})$ and Corollary 1, we know that Eq. (13) is equivalent to

$$\begin{aligned}\arg\min_{\hat{\mathbf{u}},\xi,b}\quad & \frac{1}{2}\hat{\mathbf{u}}^T\mathbf{D}\hat{\mathbf{u}}+C\sum_{i=1}^{l}\xi_i\\ \text{s.t.}\quad & y_i(\hat{\mathbf{u}}^T\hat{\mathbf{f}}_i+b)\geq 1-\xi_i\\ & \xi_i\geq 0\quad \text{for } i=1,2,\ldots,l.\end{aligned}\tag{15}$$

Denote

$$\mathbf{u}=\mathbf{D}^{1/2}\hat{\mathbf{u}}\quad \mathbf{f}_i=\mathbf{D}^{-1/2}\hat{\mathbf{f}}\tag{16}$$

Then, Eq. (15) becomes

$$\begin{aligned}\arg\min_{\mathbf{u},b}\quad & \frac{1}{2}\mathbf{u}^T\mathbf{u}+C\sum_{i=1}^{l}\xi_i\\ \text{s.t.}\quad & y_i(\mathbf{u}^T\mathbf{f}_i+b)\geq 1-\xi_i\\ & \xi_i\geq 0\quad \text{for } i=1,2,\ldots,l.\end{aligned}\tag{17}$$

Comparing the formulation in Eq. (17) with that in Eq. (3), we know the problem in Eq. (17) amounts to a standard linear SVM problem when $\mathbf{V}$ is fixed. More concretely, we solve the problem in Eq. (17) by formulating the Lagrangian as follows:

$$\mathcal{L}=\frac{1}{2}\mathbf{u}^T\mathbf{u}+C\sum_{i=1}^{l}\xi_i-\sum_{i=1}^{l}\alpha_i(y_i(\mathbf{u}^T\mathbf{f}_i+b)-1+\xi_i)-\sum_{i=1}^{l}\beta_i\xi_i\tag{18}$$

where $\alpha_i\geq 0$ and $\beta_i\geq 0$ are the Lagrangian multipliers.

Recall the basic idea of dual theorem [33], we can reformulate the problem in Eq. (17) as

$$\min_{\mathbf{u},b}\max_{\alpha_i\geq 0,\beta_i\geq 0}\left(\frac{1}{2}\mathbf{u}^T\mathbf{u}+C\sum_{i=1}^{l}\xi_i-\sum_{i=1}^{l}\alpha_i(y_i(\mathbf{u}^T\mathbf{f}_i+b)-1+\xi_i)-\sum_{i=1}^{l}\beta_i\xi_i\right).\tag{19}$$

Its dual form is

$$\max_{\alpha_i\geq 0,\beta_i\geq 0}\min_{\mathbf{u},b}\left(\frac{1}{2}\mathbf{u}^T\mathbf{u}+C\sum_{i=1}^{l}\xi_i-\sum_{i=1}^{l}\alpha_i(y_i(\mathbf{u}^T\mathbf{f}_i+b)-1+\xi_i)-\sum_{i=1}^{l}\beta_i\xi_i\right).\tag{20}$$

By minimizing $\mathcal{L}$ in Eq. (20) with respect to $\mathbf{u}$, b and ξ and taking the derivative of $\mathcal{L}$ with respect to $\mathbf{u}$, b and ξ, we have

$$\frac{\partial\mathcal{L}}{\partial\mathbf{u}}=\mathbf{u}-\sum_{i=1}^{l}\alpha_i y_i\mathbf{f}_i=0,\frac{\partial\mathcal{L}}{\partial b}=\sum_{i=1}^{l}\alpha_i y_i=0,\frac{\partial\mathcal{L}}{\partial\xi_i}=C-\alpha_i-\beta_i=0.\tag{21}$$

Plus Eq. (21) back to Eq. (20) and take some simple deduction, we have the following dual optimization problem

$$\arg\max_{\alpha_i} \mathcal{G} = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{f}_i^T \mathbf{f}_j \quad \text{s.t.} \quad \sum_{i=1}^{l} \alpha_i y_i = 0$$
$$0 \le \alpha_i \le C \quad \text{for } i = 1, 2, \ldots, l. \tag{22}$$

As in traditional linear SVM, the optimization problem in Eq. (22) can be solved in an effective way, although it is still a quadratic programming [32].

In summary, when we fix $\{\mathbf{v}_i\}_{i=1}^{k}$, the solution of MRMLSVM can be computed by solving the optimization problem in Eq. (22) directly. In other words, we can regard it as taking the data, represented by $\{\mathbf{f}_i\}_{i=1}^{l}$, as the input of original linear SVM and compute the corresponding projection vectors. In Section 4, we will show that when $\mathbf{V}$ is fixed, our derived results are the global solutions to the problem in Eq. (11). Besides, as seen from above deduction, the dimensionality of $\mathbf{f}_i$ is mk. It is less than the vectorization of $\mathbf{X}_i$, which has the dimensionality mn. Thus, the computational cost is also reduced.

(2) *Fixing* $\mathbf{U}$ *and optimizing* $\mathbf{V}$ *and* b. Similarly, when $\mathbf{U}$ is fixed, we can also change the formulation of our algorithm in Eq. (11) and derive its optimal $\mathbf{V}$ and b by solving another linear SVM problem. More concretely, Denote

$$\hat{\mathbf{g}}_i = \begin{bmatrix} \mathbf{X}_i^T \mathbf{u}_1 \\ \mathbf{X}_i^T \mathbf{u}_2 \\ \vdots \\ \mathbf{X}_i^T \mathbf{u}_k \end{bmatrix}_{nk \times 1}, \quad \hat{\mathbf{v}} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_k \end{bmatrix}_{nk \times 1}, \tag{23}$$

$$\mathbf{Q} = (\mathbf{U}^T \mathbf{U}) \otimes \mathbf{I}_{n \times n} \tag{24}$$

We can reformulate Eq. (11) as the following form:

$$\arg\min_{\hat{\mathbf{v}}, \xi, b} \quad \frac{1}{2} \hat{\mathbf{v}}^T \mathbf{Q} \hat{\mathbf{v}} + C \sum_{i=1}^{l} \xi_i$$
$$\text{s.t.} \quad y_i (\hat{\mathbf{v}}^T \hat{\mathbf{g}}_i + b) \ge 1 - \xi_i$$
$$\xi_i \ge 0 \quad \text{for } i = 1, 2, \ldots, l. \tag{25}$$

Denote

$$\mathbf{v} = \mathbf{Q}^{1/2} \hat{\mathbf{v}} \quad \mathbf{g}_i = \mathbf{Q}^{-1/2} \hat{\mathbf{g}} \tag{26}$$

Then, Eq. (25) becomes

$$\arg\min_{\mathbf{v}, \xi, b} \quad \frac{1}{2} \mathbf{v}^T \mathbf{v} + C \sum_{i=1}^{l} \xi_i \quad \text{s.t.} \quad y_i (\mathbf{v}^T \mathbf{g}_i + b) \ge 1 - \xi_i$$
$$\xi_i \ge 0 \quad \text{for } i = 1, 2, \ldots, l. \tag{27}$$

The problem in Eq. (27) amounts to a standard SVM problem, taking $\{\mathbf{g}_i\}$ as the input. We can also adopt the same method as previous to derive the global optimization to this problem. The dimensionality of input is nk. It is also less than mn. Due to the limitation of space, we omit the details.

In summary, as seen from above procedure, when we fix one parameter and optimize the other, we can derive the solutions by employing traditional SVM on two different data sets with different formulations.

There are several points should be highlighted here as follows:

(1) The first is about initialization. As seen from above procedure, MRMLSVM is solved in an iterative way. Thus, initialization is vital for searching optimal solution. In next section, we will provide three different kinds of initialization methods, i.e., fixed, uniform and norm. See more details there.
(2) The second is about the convergence behavior of our algorithm. As seen from the problem in Eq. (11), our formulation is not joint convex with respect to $\mathbf{U}$, $\mathbf{V}$, b. Thus, it is solved in an alternative way. In next section, we will show that the objective function in Eq. (11) will decrease through this kind of iteration. In other words, the above iteration is convergent to a local optimization to the problem in Eq. (11). Our experiments also show that this kind of iteration converges fast. It is often less than ten times in our experiments.
(3) The third problem is about the extension to any order tensors. As seen from the above deduction, extension to any order is direct and the formulated problem can also be solved in similar way. For example, we can fix any two projection matrices and compute the rest one if the data is cubic.
(4) Although the final computed projecting vectors are $\mathbf{u}$ and $\mathbf{v}$, not $\mathbf{U}$ and $\mathbf{V}$, they are just different arrangements of similar vectors. We can use Eqs. (14), (16), (23) and (26) to derive $\mathbf{U}$ and $\mathbf{V}$ directly. Besides, in above deductions, b and $\boldsymbol{\xi}$ are updated two times in each iteration. We have not distinguished them in deriving $\mathbf{U}$ and $\mathbf{V}$.

In summary, the procedure of MRMLSVM is listed in Table 2.

4. Performance analysis

In this section, we will analyze our proposed algorithm in four different aspects, including convergence behavior, initialization problem, computational complexity and parameter determination.

4.1. Convergence analysis

As seen from the procedure in Table 2, we solve the optimization problem in an alternative way. Namely, we fix one variable and compute the other. When $\mathbf{U}$ is fixed, the following proposition shows that we can find the global optimization to the problem in Eq. (12). Similarly, the derived results of the problem in Eq. (27) are also the solutions to the problem in Eq. (11) if $\mathbf{U}$ is fixed.

Proposition 3. *When* V *is fixed, the results* $\mathbf{U}$ *and* b*, derived by solving problem in Eq. (17), are the global solutions to the problem in Eq. (11). Similarly, when* $\mathbf{U}$ *is fixed, the results derived by solving problem in Eq. (27) are also the global solutions to the problem in Eq. (11).*

The proof is direct. Briefly, when $\mathbf{V}$ is fixed, the optimization problem in Eq. (11) is equivalent to the problem in Eq. (17). Considering the procedure of traditional SVM, we can derive its global optimization by solving the dual problem in Eq. (22). Thus, the solution to Eq. (17) is also the global optimization to the problem in Eq. (11), provided that $\mathbf{V}$ is fixed. Similarly, we can also conduct the same conclusion when $\mathbf{U}$ is fixed. Intuitively, when one parameter is fixed, we will compress a matrix into one direction (row or column). MRMLSVM could find the best decision boundary to classify the compressed vector data.

The above proposition only shows that the proposed algorithm could find a global optimization in each iteration. Based on this result, we will propose another proposition. It indicates that our iteration in Table 2 will find the solutions, which satisfy the constraint in Eq. (11) and monotonically decrease the objective function of the problem in Eq. (11).

Proposition 4. *The iterative procedure shown in Table 2 will find the solution that satisfies the constraints in Eq. (11) and monotonically decreases the objective function of the problem in Eq. (11) in each iteration.*

The proof is listed in Appendix. This proposition indicates that we can find a local optimal solution to the problem in Eq. (11). Besides, the final results are closely related to initialization. If we have initialized suitably, our derived solution will be near to the optimal solution. We will discuss the initialization problem in next section. Experimental results also show that this kind of iteration is effective. It converges fast and the time of iteration is often less than 10.

Table 2
Procedure of MRMLSVM.

Training process:
Input: Training matrix data set: $\{\mathbf{X}_i|i=1,2,\ldots,l\}$, label vectors: $\{y_i|i=1,2,\cdots,l\}$
Output: Left and right projection vectors: $\mathbf{U}$, $\mathbf{V}$, b

1. Initialize $\mathbf{V}$ by one of the three strategies shown in Section 4.2 and formulate $\mathbf{f}_i$ in Eq. (16)
2. Alternatively update $\mathbf{u}$ and $\mathbf{v}$ until convergence
   a. Update $\mathbf{u}$ and b by solving the problem in Eq. (17), where $\mathbf{D}$ and $\mathbf{f}_i$ in Eqs. (14) and (16) are computed based on the latest $\mathbf{V}$
   b. Update $\mathbf{v}$ and b by solving the problem in Eq. (27), where $\mathbf{Q}$ and $\mathbf{g}_i$ in Eqs. (24) and (26) are computed based on the latest $\mathbf{U}$
3. Reformulate $\mathbf{u}$ and $\mathbf{v}$ to $\mathbf{U}$ and $\mathbf{V}$ according to Eqs. (14), (16), (23) and (26)

Testing process:
Input: Testing matrix data set: $\{\mathbf{X}_i|i=l+1,l+2,\ldots,l+t\}$. The left and right projection vectors $\mathbf{U}$, $\mathbf{V}$ and b
Output: The labels of testing data: $\{y_i|i=l+1,l+2,\ldots,l+t\}$

1. Compute the decision values of $\mathbf{X}_i$: $\mathrm{Tr}(\mathbf{U}^T\mathbf{X}_i\mathbf{V})+b$
2. Assign the label of $\mathbf{X}_i$, i.e., y_i for $i=l+1,l+2,\ldots,l+t$

$$y_i=\begin{cases}1, & \mathrm{Tr}(\mathbf{U}^T\mathbf{X}_i\mathbf{V})+b>0\\ -1, & \mathrm{Tr}(\mathbf{U}^T\mathbf{X}_i\mathbf{V})+b<0\end{cases}$$

4.2. Initialization

Since our method is solved in an alternative way, the final solution has close relationship with initialization. We would like to introduce three different kinds of initialization strategies in this section.

The first initialization is an empirical method. Since 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, we can initialize MRMLSVM in a same way, i.e., $\mathbf{V}=[\mathbf{I}_{k\times k},\mathbf{0}_{k\times(n-k)}]^T$. For convenience, we call this kind of initialization 'Fixed' in the following. As seen from the following results, although this kind of initialization is simple, it is effective. In the following experiments, we will use this kind of initialization without specification.

The second kind of initialization is random. We generate some random elements, which is sampled from a uniform distribution (range from 0 to 1) to form the matrixes with corresponding sizes. In other words, we formulate $\mathbf{R}\in\mathbb{R}^{n\times l_2}$ for 2DLDA, $\mathbf{v}\in\mathbb{R}^n$ for STM and $\mathbf{V}\in\mathbb{R}^{n\times k}$ for MRMLSVM by these elements. Intuitively, each column vector of the initialization matrix takes the role of weighting the column vectors of $\mathbf{X}$. In the following, it is named as 'Uniform' for convenience.

To show the influence of different sampling methods, we generate a random matrix, whose elements are sampled from another distribution, i.e., standard normal distribution. It is named as 'Norm' in the following. In next section, we will provide some experimental results to compare different kinds of initializations. See more details in Section 5.3.

Finally, as mentioned above, we can also use the solution of STM to initialize MRMLSVM by simply setting $\mathbf{v}_1$ as STM's solution and others as not all zeros. By using this kind of initialization, we can guarantee that the training error of MRMLSVM is no larger than that of STM. The objective function values in using this initialization are reported in Section 5.2.

4.3. Computational complexity

We compare the computational complexity of different methods, including linear SVM, LDA, 2DLDA, STM and MRMLSVM, in this section. Since different implementations of the same method may cost different time, we would like to give a common analysis merely.

The first group of methods contains LDA and 2DLDA. Considering the procedure of 2DLDA in Section 2.2, we can see that the most time consuming step is the eigen-decomposition. If the dimensionality of original data is D, its computational complexity is about $O(D^3)$. Thus, the computational complexity of traditional LDA is $O(m^3n^3)$. Since 2DLDA solves two eigen-decomposition problem with the sizes m and n iteratively. It has the computational complexity $O(s(m^3+n^3))$, where s is the number of iterations.

The second group of methods consists of SVM, STM and MRMLSVM. The most time consuming step of these methods is the formulated QP problem in Eq. (21). Since different implementations of the same method may cost different time, it has the computational complexity about $O(slD)$ in one famous implementation LibSvm [9]. Here D is also the dimensionality of input and l is the number of training points. s is also the times of iterations. When SVM is employed, its computational complexity is $O(slmn)$. By contrast, since MRMLSVM is solved by an alternation between two different SVMs, its computational complexity is $O(sl(m+n)k)$. Since STM is a special case of MRMLSVM with $k=1$, its computational complexity is $O(sl(m+n))$.

Besides, in the implementation of MRMLSVM, it still costs a lot of time in computing the inverse and square root of matrix $\mathbf{D},\mathbf{Q}$ as shown in Eqs. (16) and (26). Recalling the above deduction, we know that their dimensionality are $mk\times mk$ and $nk\times nk$ respectively. Notice the formulations of $\mathbf{D},\mathbf{Q}$ in Eqs. (14) and (24), we can see that they are both Kronecker products of two positive semi-definite matrices. To compute its inverse and square root in a more effective way, we proposed the following proposition.

Lemma 2. *Suppose the SVD decompositions of* $\mathbf{A}$ *and* $\mathbf{B}$ *are* $\mathbf{A}=\mathbf{U}_A\mathbf{\Sigma}_A\mathbf{V}_A^T$ *and* $\mathbf{B}=\mathbf{U}_B\mathbf{\Sigma}_B\mathbf{V}_B^T$. *Then*

$$\mathbf{A}\otimes\mathbf{B}=(\mathbf{U}_A\otimes\mathbf{U}_B)(\mathbf{\Sigma}_A\otimes\mathbf{\Sigma}_B)(\mathbf{V}_A\otimes\mathbf{V}_B)^T \tag{28}$$

We would like to omit the proof and see more details in [34].

Based on above lemma, we can get the following proposition, which facilitates the computation of $\mathbf{D}^i$ and $\mathbf{Q}^i$ for any i.

Proposition 5. *Suppose* $\mathbf{A}$, $\mathbf{B}$ *are positive semi-definite and* i *is an integer, then*

$$(\mathbf{A}\otimes\mathbf{B})^i=\mathbf{A}^i\otimes\mathbf{B}^i \tag{29}$$

The proof is listed in Appendix. Based on above proposition, we can decompose the inverse computation of an integrated

matrix $\mathbf{D}$ (formed by Kronecker product with $mk \times mk$ dimensionality) into the inverse computation of a small matrix $\mathbf{V}^T\mathbf{V}$ (with dimensionality $k \times k$). Moreover, in the above iteration, s is less than 10 and k is far less than $\min\{m, n\}$. Thus, the computational complexity of MRMLSVM is smaller than SVM, especially when the scale of matrix is large. We will show some experimental results in Section 5.5.

4.4. Parameter determination

In this section, we would like to discuss the problem of parameter determination. As seen from Eq. (11), there is one important parameter k in our method. It is the rank of regression.

Recalling the essence of k as discussed in Section 3.3, we can regard it as a parameter to balance the capacity of learning and generalization. The larger k indicates the stronger learning capacity and weaker generalization capacity and vice versa. Based on the basic rule of learning theory, these two capacities cannot be improved simultaneously. Thus, it cannot be too large or too small in our implementation. We will show some numerical results concerning objective function and classification accuracy with different k in Section 5.6.

Since parameter determination is still an open problem, we would like to determine it heuristically and empirically in our paper. One direct way is using grid search as in most unsupervised learning. More concretely, we vary this parameter within a certain range and choose the one with the best performance. Another way is using cross validation as in most supervised learning. We split all the data into several proportions and use parts of them for training and the left as testing. The parameter with best classification accuracy is selected.

Although the above mentioned strategies are effective, it is very time consuming for real applications. As seen from the experimental results shown in Figs. 5 and 6 in Section 5.6, when $k=2$, the classification accuracy of MRMLSVM is near the peak. Thus, we empirically choose $k=2$ in our following experiments.

5. Experiments

In this section, we will evaluate our method in five different aspects. The first is about the convergence behavior. The second contains the comparison of classification accuracy, including results on binary classification and multiple class scenario. In multiple class scenario, we classify the data via *one-vs-rest* strategy as in traditional methods. We will show the influence of different initialization in the third group of experiments. In the fourth group, we will compare the real time consuming of different methods. Finally, experimental results with different parameter k are presented.

Since SVM, 2DLDA and STM are closely related to MRMLSVM, we would like to compare the performance of our method with them. Additionally, we also provide the results of LDA since 2DLDA can be regarded as its two-dimensional extension. Nevertheless, LDA and 2DLDA cannot be used for classification directly. We use them for projection and employ other classifiers, such as SVM, for classification. They are denoted as LDASVM and 2DLDASVM in the following presentations. We also present the results of K-nearest neighborhood classifier (KNN) as the baseline, where $K=1$. Besides, we also compare with the methods concerning spatial information. The first method is S-LDA [14]. We use it as dimensionality reduction approach and employ SVM for classification. It is named as S-LDASVM. In the second approach, we simply add the spatial matrix in [14], i.e., $\Delta^T\Delta$, as a regularizer in SVM. We name it as spatial regularized SVM (SRSVM) and its objective function is

$$\begin{aligned} \arg\min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i + \alpha\mathbf{w}^T\Delta^T\Delta\mathbf{w} \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1-\xi_i \\ & \xi_i \geq 0 \quad \text{for } i = 1, 2, \ldots, l, \end{aligned} \tag{30}$$

where α is a balance parameter.

To solve this problem, we transform it into the formulation which can be manipulated by linear SVM. Moreover, we use the first kind of initialization strategy. The dimensionality of subspace is set as $c-1$ in LDA and 2DLDA as in traditional approaches, where c is the number of class. Without specification, we set $k=2$ and C, α are determined by five-folder cross validation.

5.1. Data description

In the following experiments, we use six different kinds of matrix data sets for evaluation. They are Coil data set,[1] Pedestrian data set,[2] Binucleate data set,[3] Umist data set,[4] FingerDB data set[5] and Pollen data set.[6] These matrix data sets are selected from different applications. The data size ranges from about 50 to 2000 and the data scale ranges from 25×25 to 1204×1280. Namely, the dimensionality of vectorized sample ranges from about 600 to 1,000,000. After some preprocessing, we select part of them for illustration and the detailed statistical characters are listed in Table 3.

5.2. Convergence behavior

In order to show the convergence behavior and compare the objective function of different methods, we provide some numerical results on three data sets. We focus on the training error in binary classification task and choose three data sets, i.e., Pedestrian data, Umist data(1 vs 18) and Pollen data (2 vs 7) as training points and report their objective function values with different initializations. After 20 times iterations, the objective function values are shown in Fig. 1. On the top and bottom planes, STM and MRMLSVM are initialized using 'Fixed' and 'Norm' strategies respectively. Besides, we also show the objective function values of SVM. When we use STM's results to initialize MRMLSVM, its results are denoted by 'MRMLSVM+STM'. In these experiments, we simply set $k=2$ in MRMLSVM.

There are mainly two observations from the results in Fig. 1:

(1) With the increase of iteration's number, the function values of STM and MRMLSVM converge to fixed points. It indicates that the above solving strategy converges. It can also validate the theoretical results shown in Proposition 4.
(2) As seen from all the sub-figures, SVM's objective function value is smallest. If we use STM to initialize MRMLSVM, its objective function value is smaller than that of STM. Besides, if we use the same initialization strategy, the objective function value of MRMLSVM is also smaller than that of STM. It also validates our previous statement that k is a parameter who can rank the objective function values.

[1] http://www1.cs.columbia.edu/CAVE/research/softlib/coil-20.html.
[2] http://www.lookingatpeople.com/download-daimler-ped-mcue-occl-class-benchmark/index.html.
[3] http://ome.grc.nia.nih.gov/iicbu2008/binucleate/index.html.
[4] http://images.ee.umist.ac.uk/danny/database.html.
[5] http://bias.csr.unibo.it/fvc2000/databases.asp.
[6] http://ome.grc.nia.nih.gov/iicbu2008/pollen/index.html.

5.3. Classification results

In binary classification task, every two classes are combined and parts of the results are presented. For illustration, we report two results (1 vs 18, 4 vs 14) on Umist data, one comparison (1 vs 8) on FingerDB data and one comparison (2 vs 7) on Pollen data. Pedestrian data and Binucleate data are also employed for binary classification. By randomly splitting the original data into training set and testing set for 50 independent runs, we select 2, 3, …, 11 samples from each category and the others are testing samples for Binucleate, Umist, Pollen data sets. Since Pedestrian has many samples and FingerDB has few points for each category, we select 2, 4, …, 20 and 2, 3, …, 7 training samples from each category from Pedestrian and FingerDB data sets. With different data sets and different numbers of training samples, the mean classification accuracies of 50 independent runs are shown in Fig. 2 (a) Pedestrian data, (b) Binucleate data, (c) Umist data (1 vs 18), (d) Umist (4 vs 14), (e) FingerDB data (1 vs 8) and (f) Pollen data (2 vs 7). Note that, since the matrix $\Delta^T\Delta$ in SRSVM and S-LDASVM is $mn \times mn$ dimensional, these methods cannot be implemented on Binucleate and FingerDB data sets. We have not reported their results.

To classify data from multiple classes, we choose four representative data, including Coil, Umist, FingerDB and Pollen data sets. As seen from the description in Table 3, they are data sets with multiple classes. In each category, we randomly selected 2, 3,…, 11 training samples for Coil, Umist, Pollen data and 2, 3,…, 7 for FingerDB data. The mean and standard derivation computed by 50 random splits are shown in Tables 4–7. The results with statistical significance are boldfaced. As the same reason in previous, we have not reported the results on FingerDB either.

There are several observations from the above comparisons as follows:

(1) Among different methods and different data sets, MRMLSVM achieves the highest accuracy in most cases. This is mainly due to the fact that MRMLSVM inherits the merits from SVM, STM and spatial regularization methods.
(2) With the increase of training points' number, all methods achieve higher accuracies. This consists with intuition since we have more prior information for training.

Table 3
Characters of different data sets.

Data	Size $(l+t)$	Scale $(m \times n)$	Class number
Coil	1440	32×32	20
Pedestrian	2000	38×18	2
Binucleate	40	1204×1280	2
Umist	575	28×23	20
FingerDB	80	300×300	10
Pollen	630	25×25	7

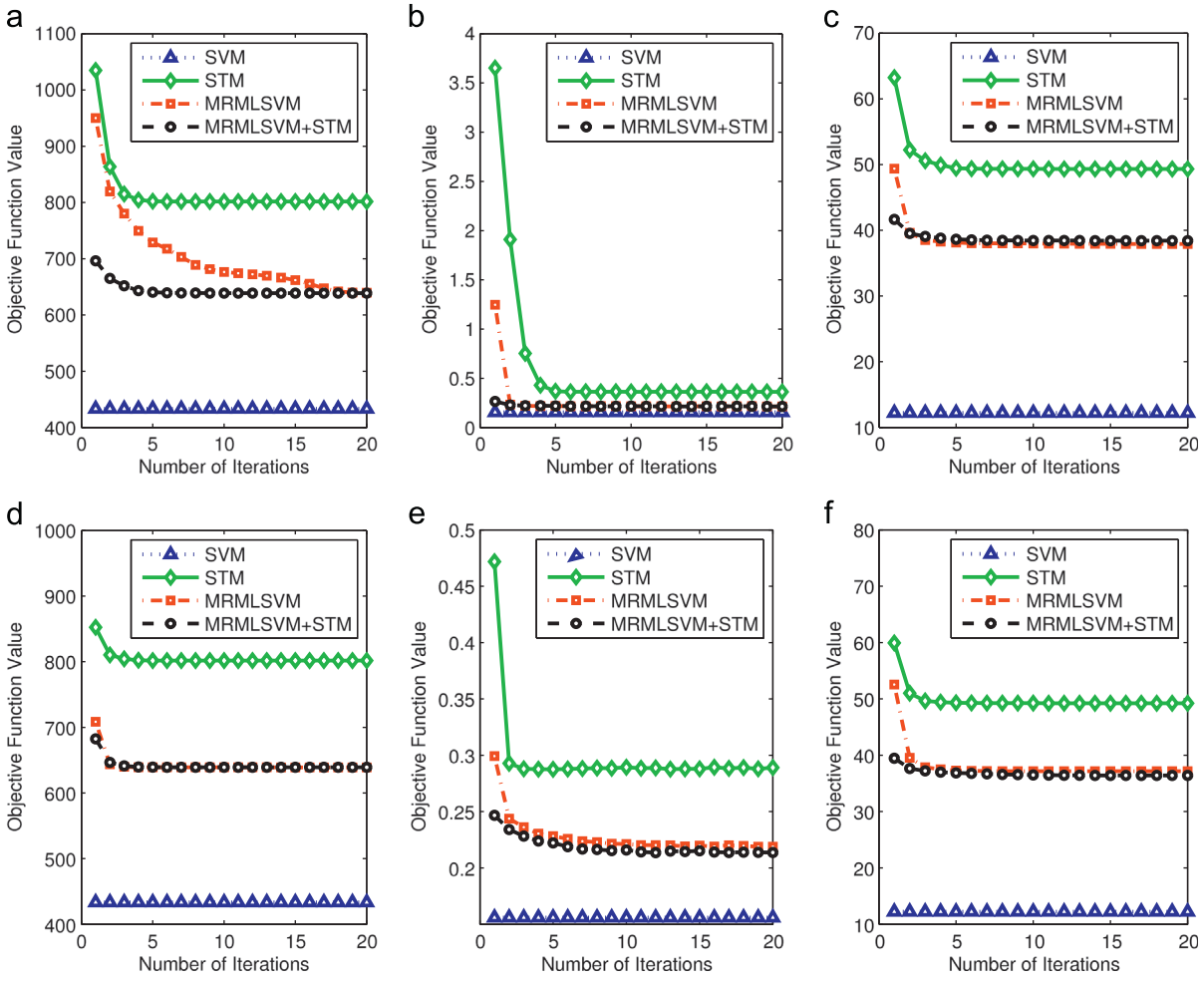

Fig. 1. Objective function values on three data sets with different initializations. The x-axis represents number of iterations. MRMLSVM+STM means that we used the results of STM to initialize MRMLSVM. (a) Pedestrian data with 'Fixed' initialization; (b) Umist data (1 vs 18) with 'Fixed' initialization; (c) Pollen data (2 vs 7) with 'Fixed' initialization; (d) Pedestrian data with 'Norm' initialization; (e) Umist data (1 vs 18) with 'Norm' initialization; (f) Pollen data (2 vs 7) with 'Norm' initialization.

(3) By adding the spatial smooth regularization, the improved methods are perform better than their original counterparts in most cases. This may be due to the fact that we have characterized the spatial information and traditional methods can be regarded as their special cases.
(4) For classification, 2D based methods do not always perform better than 1D based methods. Take the results in Table 6 as an example, LDASVM performs better than 2DLDASVM in most cases. The reason may be that the adding constraints in 2DLDASVM will degrade the performances.
(5) When we represent original data by its low-dimensional embedding, it does not always helpful for classification. Take the result in Table 6 as an example, compared with SVM's results in classifying the embedding derived by LDA and 2DLDA, SVM achieves higher accuracy when it is used to classify the original data directly.
(6) When the dimensionality of original data is high, it seems that margin based methods, such as SVM, perform better than similarity based methods, such as KNN. It is mainly due to the reason that the distance between any two high-dimensional data points trends to be similar [13] and the performance of similarity based methods, such as KNN, will be degraded in this scenario.

5.4. Different initializations

In this section, we will show the influence of different initializations. In Section 4.2 we have discussed three kinds of initializations. Briefly, the first is 'Fixed', which uses fixed value. The second is 'Uniform', which initializes randomly with a uniform distribution. The third is 'Norm', whose initialization matrix consists of elements sampled from standard normal distribution.

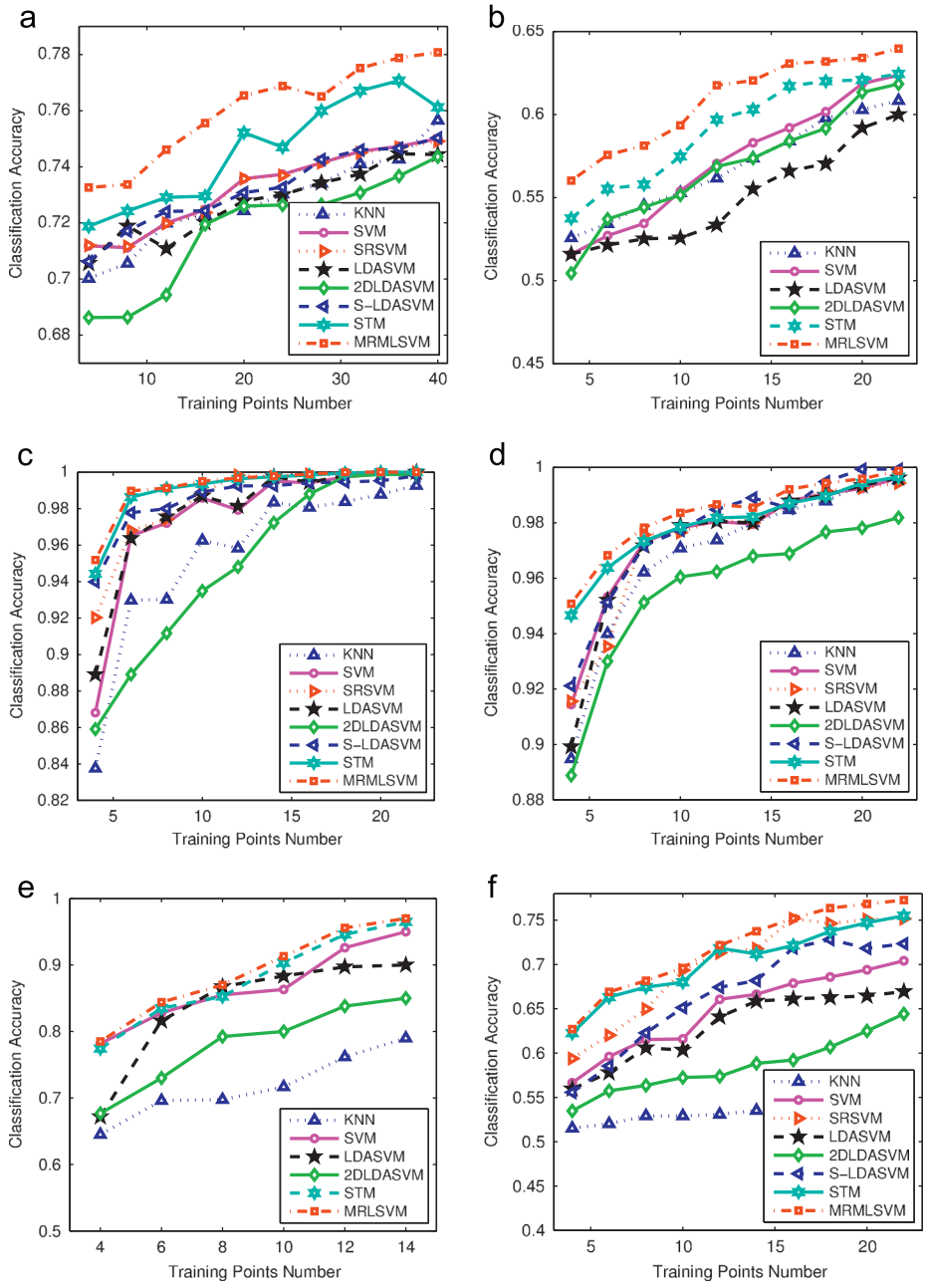

Fig. 2. Classification results of different methods on six different data sets with different numbers of training points. The number of training points varies from 4 to 22. (a) Pedestrian data; (b) Binucleate data; (c) Umist data (1 vs 18); (d) Umist (4 vs 14); (e) FingerDB data (1 vs 8); (f) Pollen data (2 vs 7).

Table 4
Classification accuracy of different methods on Coil data (mean ± std/%).

TRAINING NO.	KNN	SVM	SRSVM	LDASVM	2DLDASVM	S-LDASVM	STM	MRMLSVM
40	71.80 ± 1.30	**72.18** ± 1.38	**73.95** ± 1.33	70.60 ± 1.64	68.04 ± 2.97	70.68 ± 1.83	**73.27** ± 2.02	**75.09** ± 1.74
60	77.17 ± 1.14	79.75 ± 1.12	79.79 ± 1.14	74.25 ± 1.35	71.03 ± 2.74	75.98 ± 1.23	**79.36** ± 1.84	**82.54** ± 1.58
80	81.03 ± 0.82	83.57 ± 1.25	**84.99** ± 1.26	78.54 ± 1.54	79.24 ± 2.37	79.71 ± 1.30	83.80 ± 1.52	**86.78** ± 1.31
100	82.60 ± 0.99	85.95 ± 1.25	86.43 ± 1.29	80.37 ± 1.26	82.66 ± 2.19	81.37 ± 1.95	86.24 ± 1.42	**89.08** ± 1.13
120	85.10 ± 0.81	88.95 ± 0.89	89.57 ± 0.89	82.77 ± 1.24	86.86 ± 1.58	83.87 ± 1.18	88.56 ± 1.20	**91.79** ± 1.03
140	86.51 ± 0.89	**90.93** ± 1.21	**91.90** ± 1.19	83.91 ± 1.16	88.95 ± 1.47	85.07 ± 1.24	90.09 ± 1.31	**92.88** ± 1.13
160	87.66 ± 0.82	92.04 ± 0.93	**93.09** ± 0.93	84.86 ± 1.00	90.41 ± 1.32	85.94 ± 1.18	91.99 ± 0.87	**94.06** ± 0.75
180	88.87 ± 0.66	**93.13** ± 0.77	**94.07** ± 0.76	86.33 ± 0.81	92.17 ± 0.85	87.24 ± 0.81	92.31 ± 0.85	**94.88** ± 0.73
200	89.70 ± 0.76	**93.88** ± 0.92	**95.20** ± 0.96	86.85 ± 0.95	92.99 ± 0.98	88.15 ± 1.07	92.93 ± 0.72	**95.30** ± 0.79
220	90.96 ± 0.49	95.24 ± 0.49	95.28 ± 0.46	88.03 ± 0.68	94.06 ± 0.81	88.72 ± 0.53	93.99 ± 0.50	**96.23** ± 0.43

Table 5
Classification accuracy of different methods on Umist data (mean ± std/%).

TRAINING NO.	KNN	SVM	SRSVM	LDASVM	2DLDASVM	S-LDASVM	STM	MRMLSVM
40	60.72 ± 1.58	64.55 ± 1.53	**66.03** ± 1.48	**65.87** ± 1.32	65.57 ± 1.55	**65.95** ± 1.55	**65.98** ± 1.37	**68.32** ± 1.27
60	69.88 ± 1.15	75.97 ± 1.14	**77.76** ± 1.14	**77.62** ± 1.24	**78.61** ± 1.39	**77.87** ± 1.04	**77.70** ± 1.05	**79.06** ± 0.97
80	78.36 ± 1.04	84.94 ± 0.93	84.99 ± 0.95	**85.39** ± 0.95	**85.20** ± 1.08	**85.39** ± 0.92	84.49 ± 1.08	**87.21** ± 1.00
100	82.56 ± 1.04	88.82 ± 0.88	**89.68** ± 0.89	88.17 ± 0.98	88.32 ± 0.96	88.65 ± 0.80	87.68 ± 0.84	**90.85** ± 0.78
120	87.95 ± 0.86	**92.71** ± 0.88	**92.75** ± 0.86	91.82 ± 0.88	91.45 ± 0.74	91.82 ± 0.90	91.12 ± 0.74	**93.92** ± 0.69
140	90.92 ± 0.77	93.41 ± 0.59	93.68 ± 0.57	93.27 ± 1.04	93.85 ± 0.65	93.32 ± 0.55	93.54 ± 0.70	**95.70** ± 0.65
160	92.22 ± 0.75	94.68 ± 0.57	94.87 ± 0.58	94.43 ± 0.81	94.10 ± 0.51	94.94 ± 0.50	94.54 ± 0.64	**96.48** ± 0.59
180	93.96 ± 0.64	95.63 ± 0.44	**96.24** ± 0.43	95.15 ± 0.55	95.58 ± 0.46	95.86 ± 0.49	96.01 ± 0.47	**97.07** ± 0.44
200	95.32 ± 0.55	96.18 ± 0.52	**97.01** ± 0.55	95.84 ± 0.55	95.92 ± 0.58	**96.74** ± 0.52	96.56 ± 0.46	**97.64** ± 0.43
220	95.94 ± 0.62	96.50 ± 0.47	**97.60** ± 0.50	96.23 ± 0.67	96.22 ± 0.43	**97.34** ± 0.45	96.71 ± 0.45	**98.01** ± 0.43

Table 6
Classification accuracy of different methods on FingerDB data (mean ± std/%).

TRAINING NO.	KNN	SVM	LDASVM	2DLDASVM	STM	MRMLSVM
20	10.02 ± 1.34	49.83 ± 2.39	44.16 ± 2.77	34.75 ± 2.98	**50.89** ± 2.05	**54.91** ± 2.00
30	10.10 ± 1.05	**60.520** ± 2.28	45.40 ± 2.79	37.30 ± 2.68	59.40 ± 2.03	**64.30** ± 2.00
40	10.38 ± 1.11	64.25 ± 2.31	48.25 ± 2.27	40.25 ± 2.34	**66.50** ± 1.97	**70.50** ± 1.92
50	10.67 ± 1.18	69.00 ± 2.38	51.50 ± 2.12	45.50 ± 2.05	69.83 ± 1.92	**78.50** ± 1.87
60	11.00 ± 1.31	77.50 ± 1.70	51.75 ± 1.81	46.50 ± 1.62	73.50 ± 1.72	**82.00** ± 1.68
70	11.36 ± 1.04	78.00 ± 1.80	57.00 ± 1.62	49.50 ± 1.58	78.45 ± 1.63	**83.50** ± 1.59

Table 7
Classification accuracy of different methods on Pollen data (mean ± std/%).

TRAINING NO.	KNN	SVM	SRSVM	LDASVM	2DLDASVM	S-LDASVM	STM	MRMLSVM
14	21.70 ± 2.36	37.59 ± 2.00	**39.88** ± 1.87	32.18 ± 1.88	24.12 ± 2.06	**41.01** ± 2.14	**40.18** ± 1.68	**41.05** ± 1.87
21	21.88 ± 2.18	**42.40** ± 2.14	**43.07** ± 1.95	37.14 ± 1.88	28.41 ± 1.75	**44.35** ± 2.27	**43.62** ± 1.58	**44.86** ± 1.81
28	22.37 ± 1.96	**44.97** ± 1.93	**46.64** ± 1.91	39.26 ± 1.58	32.72 ± 1.94	**46.04** ± 1.94	**45.24** ± 1.53	**47.29** ± 1.75
35	23.16 ± 1.88	**46.65** ± 1.49	**49.77** ± 1.46	41.76 ± 1.54	35.00 ± 1.68	**48.47** ± 2.15	46.73 ± 1.50	**49.73** ± 1.72
42	23.57 ± 1.86	47.19 ± 1.33	**50.45** ± 1.26	42.68 ± 1.49	39.93 ± 1.66	**49.24** ± 1.60	**49.20** ± 1.24	**50.85** ± 1.42
49	23.64 ± 1.76	**50.81** ± 1.25	**52.74** ± 1.29	44.27 ± 1.43	41.08 ± 1.63	49.97 ± 1.45	**51.01** ± 1.20	**52.90** ± 1.37
56	24.11 ± 1.74	51.42 ± 1.12	**53.27** ± 1.17	44.95 ± 1.45	44.87 ± 1.56	50.62 ± 1.22	51.54 ± 1.11	**53.99** ± 1.27
63	24.74 ± 1.67	**53.32** ± 1.11	**55.47** ± 1.09	45.32 ± 1.34	47.66 ± 1.56	50.75 ± 1.18	**54.42** ± 1.06	**55.61** ± 1.22
70	24.83 ± 1.72	54.05 ± 1.09	**56.20** ± 0.95	45.79 ± 1.38	48.52 ± 1.48	50.84 ± 1.19	54.50 ± 0.91	**56.52** ± 1.04
77	25.37 ± 1.62	55.00 ± 1.06	**57.06** ± 1.01	46.50 ± 1.32	51.16 ± 1.43	50.91 ± 1.11	55.09 ± 0.90	**57.21** ± 1.03

We have conducted experiments on two data sets, i.e., Umist and Pollen. With fixed training points (two for each category) and testing points, we randomly initialize our method for 50 runs. Since 2DLDA and STM are also solved in an alternative way, their results are also presented. Other settings are the same as previous. For example, we also set $k=2$. The experimental results, which are the average classification accuracies of 50 runs, are shown in Fig. 3.

As seen from the results in Fig. 3, all methods have different performances with different initializations. Nevertheless, it seems that the variance between different initializations is not so significant. Thus, we have used the 'Uniform' initialization strategy in other experiments. Besides, different initializations take different influences on different data sets. For example, the variance of 2DLDASVM is larger on Pollen data than on Umist data. More importantly, as shown in Fig. 3, MRMLSVM achieves the highest

classification accuracy in almost all cases. It can also demonstrate the effectiveness

5.5. Computational complexity

Commonly, another motivation for investigating tensor based methods is reducing the computational complexity of original methods in manipulating vectorized high-dimensional data. For comparison, we will show some experimental results on several data sets with different sizes and scales.

We have selected three representative data, i.e., Pedestrian, FingerDB and Binucleate. Pedestrian data has the largest data size and Binucleate has the highest resolution. We compare MRMLSVM with SVM, SRSVM, LDASVM, 2DLDASVM, S-LDASVM, STM and use LibSvm for the realization of SVM. For justice, these methods are implemented in their original formulations, without using other accelerating strategies. When the number of training points is fixed, we randomly select them for 50 independent runs. With a naive MATLAB implementation, the calculations are conducted on a 3.2-GHz, 4G RAM Windows machine. The computational time of different methods is listed in Tables 8–10. Since LDASVM, SRSVM and S-LDASVM cannot be implemented on FingerDB and Binucleate data sets, we only reports their results on Pedestrian data. The results with statistical significance are also boldfaced.

As seen from the results in Tables 8–10, we will analyze the influence of different factors. (1) When the scale of matrix data is small and the data size is large, MRMLSVM costs similar time to SVM and consumes less time than LDASVM and 2DLDASVM. With the increase of data scale, the superiority of two-dimensional methods is emerged. For example, LDASVM, SRSVM and S-LDASVM cannot be implemented on FingerDB while 2DLDASVM still works. Compared with SVM, MRMLSVM is more suitable for dealing with large scale matrix data. Certainly, STM costs the least time. (2) The computational complexity of different methods is dominated by different factors. For example, LDASVM and 2DLDASVM are very sensitive to data scale. It is the key factor in dominating their computational complexities. Compared with dimensionality, the influence of data size is not so significant. (3) The methods with spatial regularization often cost more time than their original counterparts. This may be caused by the fact that we also need some time to compute the regularization matrix.

Table 9
Computational time of different methods with different number of training points on FingerDB data (mean ± std.). Note that LDASVM cannot be implemented in this data.

TRAINING NO.	SVM	2DLDASVM	STM	MRMLSVM
20	9.6719 ± 0.1597	35.6031 ± 0.0210	**3.6728** ± 0.4907	8.9375 ± 0.2294
30	10.0719 ± 0.2075	42.0625 ± 0.0291	**3.7321** ± 0.7846	9.0500 ± 0.2318
40	10.2906 ± 0.1975	51.4219 ± 0.0269	**3.7897** ± 0.4190	9.3812 ± 0.2203
50	10.0313 ± 0.1293	58.1313 ± 0.0259	**4.1199** ± 0.3414	9.3875 ± 0.2204
60	11.8875 ± 0.1920	65.2156 ± 0.0257	**4.3824** ± 0.3257	9.7000 ± 0.1755
70	12.3125 ± 0.1065	72.3719 ± 0.0211	**4.4481** ± 0.8383	10.2000 ± 0.1809

Table 10
Computational time of different methods with different number of training points on Binucleate data (mean ± std.). Note that LDASVM cannot be implemented in this data.

TRAINING NO.	SVM	2DLDASVM	STM	MRMLSVM
4	3.5469 ± 0.7562	1079 ± 13.2386	**0.4011** ± 0.0364	0.8854 ± 0.0203
6	6.9531 ± 0.3461	1114 ± 3.9416	**0.5330** ± 0.0345	1.2185 ± 0.0287
8	7.4687 ± 0.2923	1164 ± 2.9812	**0.6340** ± 0.0391	1.5312 ± 0.0203
10	8.0356 ± 0.2413	1172 ± 1.7536	**0.7624** ± 0.0311	1.8645 ± 0.0204
12	8.6406 ± 0.1445	1227 ± 1.4561	**0.8933** ± 0.0361	2.2083 ± 0.0212

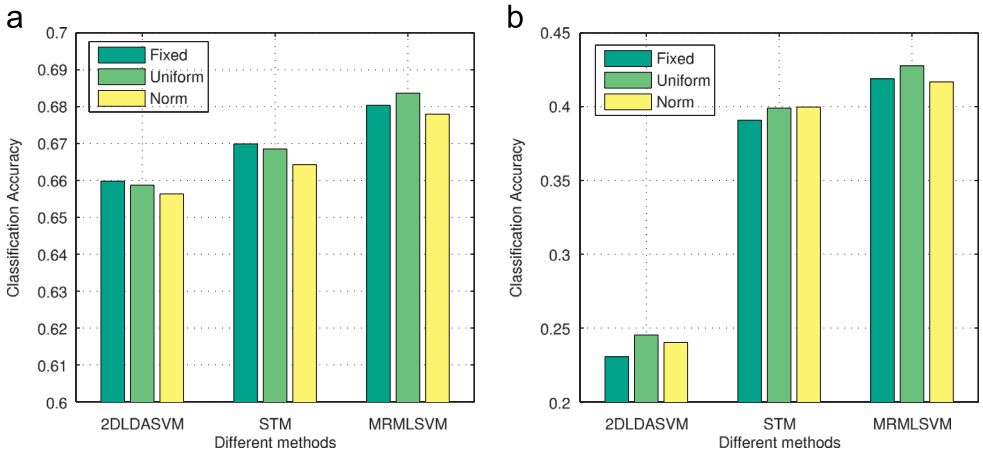

Fig. 3. The classification accuracy of 2DLDA, STM and MRMLSVM with different kinds of initialization, i.e., the fixed, uniform and norm strategies. The *x*-axis represents different methods and *y*-axis is the classification accuracy. (a) Umist data; (b) Pollen data.

Table 8
Computational time of different methods with different number of training points on Pedestrian data (mean ± std.).

TRAINING NO.	SVM	SRSVM	LDASVM	2DLDASVM	S-LDASVM	STM	MRMLSVM
200	0.3484 ± 0.0338	0.7953 ± 0.0354	2.2234 ± 0.0589	0.6438 ± 0.0264	2.3172 ± 0.2291	**0.1910** ± 0.0877	0.4531 ± 0.0516
600	0.8172 ± 0.0478	1.3672 ± 0.0488	2.3359 ± 0.0467	1.9859 ± 0.1852	2.4078 ± 0.1884	**0.3977** ± 0.0249	1.0281 ± 0.1346
1000	1.4047 ± 0.0627	1.8891 ± 0.0601	2.4281 ± 0.0331	3.1516 ± 0.4558	2.5531 ± 0.1169	**0.5986** ± 0.0284	1.9813 ± 0.0628
1400	2.2594 ± 0.0715	2.7344 ± 0.0667	2.5656 ± 0.0395	4.2516 ± 0.4382	2.6313 ± 0.1423	**1.0795** ± 0.1062	3.8438 ± 0.2300
1800	2.9922 ± 0.0773	3.4016 ± 0.0749	2.6703 ± 0.0379	5.4188 ± 0.7376	2.7360 ± 0.1393	**1.4793** ± 0.0498	5.2156 ± 0.1212

5.6. Parameter determination

As seen from the discussion in Sections 3.2 and 4.4, we know that k plays an essential role in the proposed method. In this section, we would like to show its influence in three aspects.

In the first group of experiments, we would like to show the influence of k towards objective function values. As mentioned in Proposition 2, the objective function value of MRMLSVM should monotonously decreases with respect to k, provided that we initialize it in a suitable way. For demonstration, we conduct experiments on three data sets, including Pedestrian, Umist and Pollen. The parameter k is varied from 1 to 10. We choose 2 and 10 training points from each category. With the 'Fixed' initialization, each experiment is repeated for 50 independent runs and the average objective function values are shown in Fig. 4.

As seen from the results in Fig. 4, we can conclude that with the increase of k, the average objective function values decrease monotonously. It validates the intrinsic of k in dominating training error.

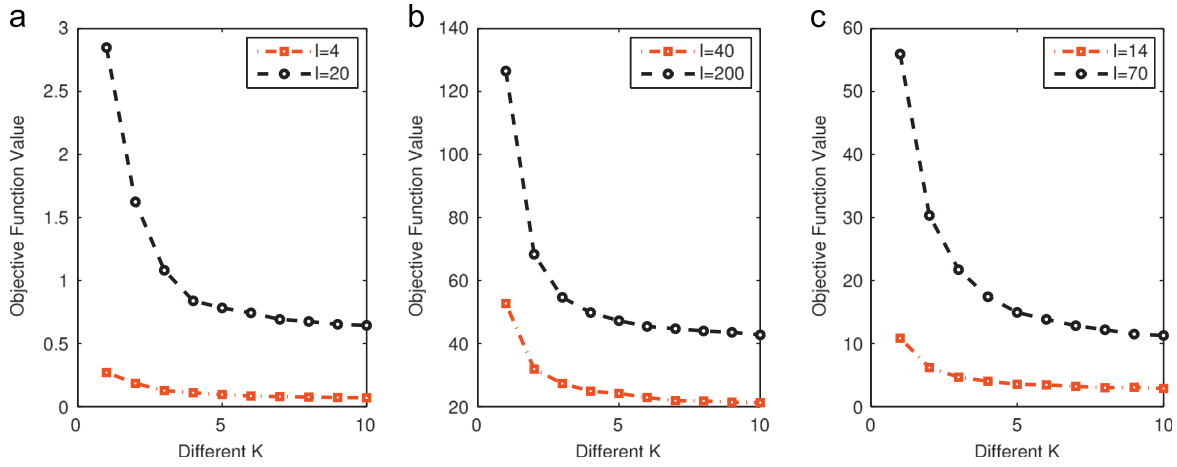

Fig. 4. The objective function of MRMLSVM with different k. The x-axis represents the selected k and y-axis is the objective function. The lines with different colors represent different number of training points. (a) Pedestrian data; (b) Umist data; (c) Pollen data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

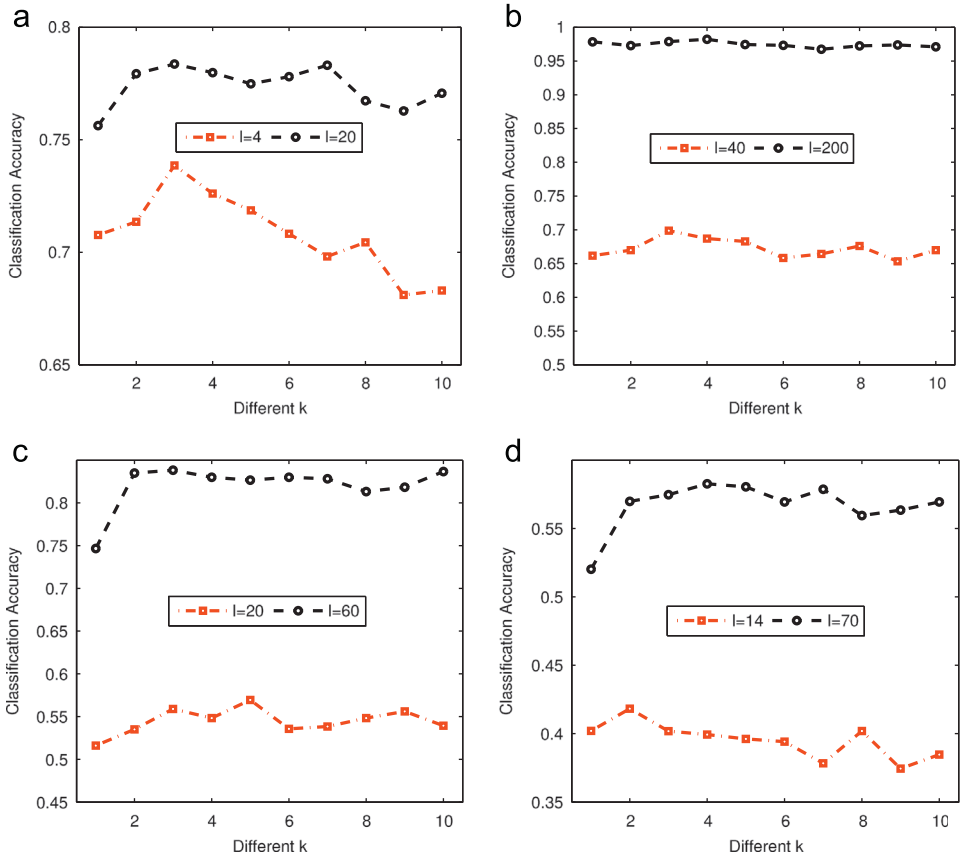

Fig. 5. The classification accuracy of MRMLSVM with different k. The x-axis represents the selected k and y-axis is the classification accuracy. The lines with different colors represent different number of training points. (a) Pedestrian data; (b) Umist data; (c) FingerDB data; (d) Pollen data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

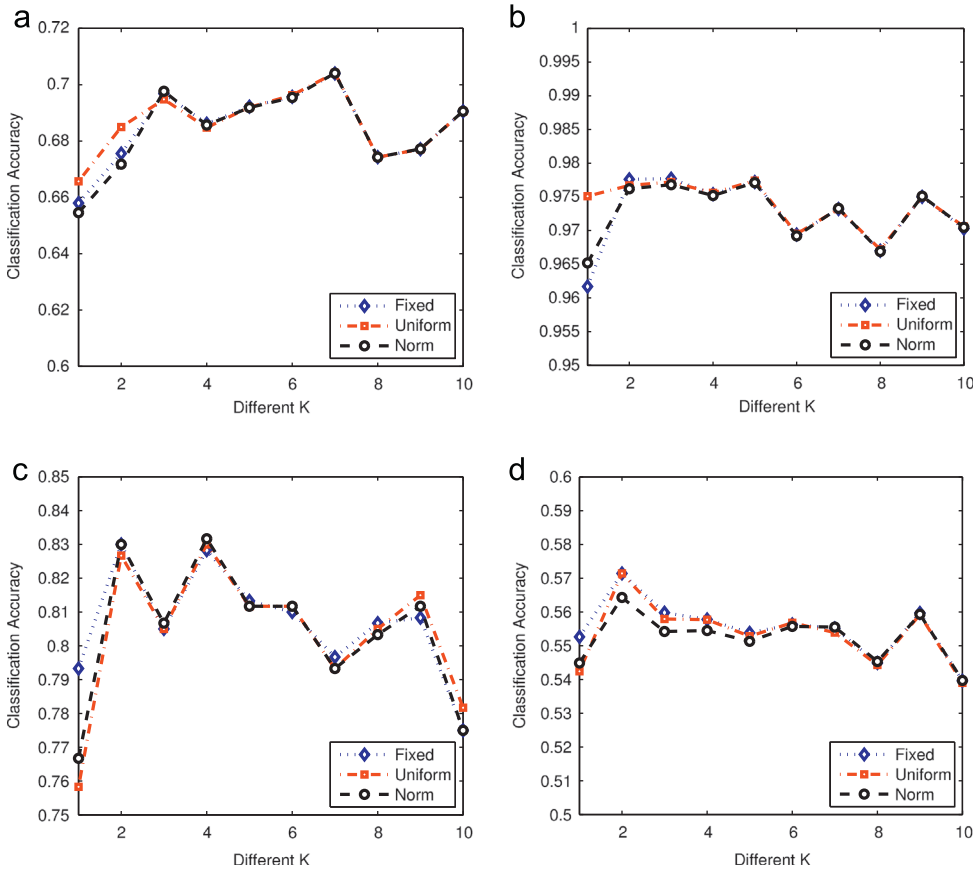

Fig. 6. The classification accuracy of MRMLSVM with different k and initialization. The x-axis represents the selected k and y-axis is the classification accuracy. The lines with different colors represent different kinds of initialization. (a) Pedestrian data; (b) Umist data; (c) FingerDB data; (d) Pollen data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the second group of experiments, we would like to show the influence of k towards classification accuracy. There are totally four different data sets, including Pedestrian, Umist and FingerDB and Pollen. We vary k from 1 to 10 and conduct experiments using 'Fixed' initialization with different number of training points. After 50 independent splitting training and testing points, we provide the average classification accuracy of MRMLSVM in Fig. 5.

As shown in Fig. 5, we can see that with the increase of k, the feasible region is expanded. 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 training error and the extent in avoiding over fitting. These two factors are important in dominating the performance of a learning machine. SVM and STM can be regarded as two extreme cases. Additionally, these results also validate the effectiveness in employing multiple transformation vectors.

In the third group of experiments, we would like to show whether the out-performance of MRMLSVM depends on the combination of k and initialization strategies. We employ the same data sets as in previous experiments. With different k and different initialization strategies, we report the average classification accuracy by randomly selecting 10 points in each category as training samples and others are assigned as testing sets. With 50 independent runs, the average classification accuracies are shown in Fig. 6.

As seen from the results in Fig. 6, we have the following intuitions. Compared with the influences of initialization strategy, k plays a more important role in dominating the final classification accuracy. The reason may be (1) k is a parameter who can determine the learning capacity of our method and (2) when k is suitable, our solving strategy tends to find a good local optimum, no matter which initialization strategy we have selected.

6. Conclusion

In this paper, we have proposed an efficient multiple rank multi-linear SVM, i.e., MRMLSVM, for matrix data classification. Different from linear SVM which reformulated matrix data into a vector and STM which only used one left and one right projection vector, we have used several left projecting vectors and the same number of right projecting vectors to formulate objective function and construct constraints. The convergence behavior, initialization strategy, computational complexity and parameter determination problems were also analyzed. Plenty of experimental results have been proposed for illustration. Further research includes the extension of MRMLSVM to nonlinear cases. We will also focus on the accelerating issue of our algorithm.

Conflict of interest

None declared.

Acknowledgments

Supported by 973 Program(2013CB329503) and NSFC (Grant No. 91120301, 61005003, 61075004 and 61021063).

Appendix A

Proof of Proposition 1. We would like to prove it in two sides. 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 $\mathrm{rank}(\mathbf{Z})\le\min(m,n)$, 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, $\mathbf{z}=\mathrm{Vec}(\sum_i^k\mathbf{u}_i\mathbf{v}_i^T)$.

On the other hand, for any two groups of vectors, denoted as $\{\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_k\}$ and $\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k\}$, $\mathrm{Vec}(\sum_i^k\mathbf{u}_i\mathbf{v}_i^T)\in\mathbb{R}^{mn}$. Combining the above two results, we get the conclusion. □

Proof of Proposition 2. Assume that the objective function of STM with the projecting vectors $\mathbf{u}$ and $\mathbf{v}$ is $f(\mathbf{u},\mathbf{v})$. Correspondingly, the objective function of MRMLSVM with projecting vectors $\{\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_k\}$ and $\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k\}$ is $g(\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_k,\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k)$. Denote the optimal value of STM as $f(\mathbf{u}^*,\mathbf{v}^*)$. Note that

$$f(\mathbf{u}^*,\mathbf{v}^*)=g(\mathbf{u}^*,\mathbf{0},\ldots,\mathbf{0}_k,\mathbf{v}^*,\mathbf{v}_2,\ldots,\mathbf{v}_k).$$

As seen from the following results in Proposition 3, In each iteration, the objective function of MRMLSVM is not increase. Thus, the optimal function value of MRMLSVM is no larger than that of STM. □

Proof of Corollary 1. The first equation is obvious and we would like to prove the second one.

Based on Lemma 1, we have

$$\begin{aligned}\mathrm{Tr}(\mathbf{U}^T\mathbf{U}\mathbf{V}^T\mathbf{V})&=(\mathrm{Vec}(\mathbf{U}))^T\mathrm{Vec}(\mathbf{U}\mathbf{V}^T\mathbf{V})\\&=(\mathrm{Vec}(\mathbf{U}))^T\mathrm{Vec}(\mathbf{I}_{m\times m}\mathbf{U}\mathbf{V}^T\mathbf{V})=\mathrm{Vec}(\mathbf{U})^T((\mathbf{V}^T\mathbf{V})\otimes\mathbf{I}_{m\times m})\mathrm{Vec}(\mathbf{U}).\end{aligned}$$

Thus, the result follows. □

Proof of Proposition 4. Assume that we have derived $\mathbf{U}^{(s)}$ in the s-th iteration. We now update $\mathbf{V}$, $\boldsymbol{\xi}$ and b. The following results hold:

$$\begin{aligned}&\{\mathbf{V}^{(s)},\boldsymbol{\xi}^{(s)},b^{(s)}\}=\arg\min_{\mathbf{V},b,\boldsymbol{\xi}}\frac{1}{2}\mathrm{Tr}\left(\mathbf{U}^{(s)}\mathbf{V}^T\mathbf{V}(\mathbf{U}^{(s)})^T\right)+C\sum_{i=1}^{l}\xi_i\\&\text{s.t.}\quad y_i(\mathrm{Tr}((\mathbf{U}^{(s)})^T\mathbf{X}_i\mathbf{V})+b)\ge1-\xi_i\\&\xi_i\ge0\quad\text{for } i=1,2,\ldots,l.\end{aligned}\tag{31}$$

In the $(s+1)$−th iteration, we fix $\mathbf{V}$ as $\mathbf{V}^{(s)}$ and optimize $\mathbf{U}$, $\boldsymbol{\xi}$ and b by solving the problem in Eq. (11). We have the following results:

$$\begin{aligned}&\{\mathbf{U}^{(s+1)},\boldsymbol{\xi}^{(s+1)},b^{(s+1)}\}=\arg\min_{\mathbf{U},\boldsymbol{\xi},b}\frac{1}{2}\mathrm{Tr}\left(\mathbf{U}(\mathbf{V}^{(s)})^T\mathbf{V}^{(s)}\mathbf{U}^T\right)+C\sum_{i=1}^{l}\xi_i\\&\text{s.t.}\quad y_i(\mathrm{Tr}(\mathbf{U}^T\mathbf{X}_i\mathbf{V}^{(s)})+b)\ge1-\xi_i\\&\xi_i\ge0\quad\text{for } i=1,2,\ldots,l.\end{aligned}\tag{32}$$

Similarly, when we fix $\mathbf{U}$ as $\mathbf{U}^{(s+1)}$ and optimize $\mathbf{V}$, $\boldsymbol{\xi}$ and b by solving the problem in Eq. (11), the following result holds:

$$\begin{aligned}&\{\mathbf{V}^{(s+1)},\boldsymbol{\xi}^{(s+1)},b^{(s+1)}\}=\arg\min_{\mathbf{V},\boldsymbol{\xi},b}\frac{1}{2}\mathrm{Tr}\left(\mathbf{U}^{(s+1)}\mathbf{V}^T\mathbf{V}(\mathbf{U}^{(s+1)})^T\right)+C\sum_{i=1}^{l}\xi_i\\&\text{s.t. } y_i(\mathrm{Tr}((\mathbf{U}^{(s+1)})^T\mathbf{X}_i\mathbf{V})+b)\ge1-\xi_i\ \ \xi_i\ge0\quad\text{for } i=1,2,\ldots,l.\end{aligned}\tag{33}$$

Combining the results in Eqs. (30) and (32), we know that $\{\mathbf{U}^{(s)},\mathbf{V}^{(s)},\boldsymbol{\xi}^{(s+1)},b^{(s)}\}$ and $\{\mathbf{U}^{(s+1)},\mathbf{V}^{(s+1)},\boldsymbol{\xi}^{(s+1)},b^{(s+1)}\}$ are both feasible solutions to the problem in Eq. (11). More importantly, recalling the results in Proposition 3, we have the following inequality:

$$\begin{aligned}&\frac{1}{2}\mathrm{Tr}\left(\mathbf{U}^{(s+1)}(\mathbf{V}^{(s+1)})^T\mathbf{V}^{(s+1)}(\mathbf{U}^{(s+1)})^T\right)+C\sum_{i=1}^{l}\xi_i^{(s+1)}\\&\le\frac{1}{2}\mathrm{Tr}\left(\mathbf{U}^{(s+1)}(\mathbf{V}^{(s)})^T\mathbf{V}^{(s)}(\mathbf{U}^{(s+1)})^T\right)+C\sum_{i=1}^{l}\xi_i^{(s+1)}\\&\le\frac{1}{2}\mathrm{Tr}\left(\mathbf{U}^{(s)}(\mathbf{V}^{(s)})^T\mathbf{V}^{(s)}(\mathbf{U}^{(s)})^T\right)+C\sum_{i=1}^{l}\xi_i^{(s)}\end{aligned}\tag{34}$$

It indicates the decrease of objective function during iteration. □

Proof of Proposition 5. Since $\mathbf{A}$, $\mathbf{B}$ are positive semi-definite, then, their SVD decompositions are $\mathbf{A}=\mathbf{U}_A\boldsymbol{\Sigma}_A\mathbf{U}_A^T$ and $\mathbf{B}=\mathbf{U}_B\boldsymbol{\Sigma}_B\mathbf{U}_B^T$. Thus

$$\begin{aligned}\mathbf{A}^i\otimes\mathbf{B}^i&=(\mathbf{U}_A\boldsymbol{\Sigma}_A^i\mathbf{U}_A^T)\otimes(\mathbf{U}_B\boldsymbol{\Sigma}_B^i\mathbf{U}_B^T)=(\mathbf{U}_A\otimes\mathbf{U}_B)(\boldsymbol{\Sigma}_A^i\otimes\boldsymbol{\Sigma}_B^i)(\mathbf{U}_A\otimes\mathbf{U}_B)^T\\&=(\mathbf{U}_A\otimes\mathbf{U}_B)(\boldsymbol{\Sigma}_A\otimes\boldsymbol{\Sigma}_B)^i(\mathbf{U}_A\otimes\mathbf{U}_B)^T=(\mathbf{A}\otimes\mathbf{B})^i.\ \square\end{aligned}\tag{35}$$

References

[1] J. Yang, D. Zhang, A.F. Frangi, J. Yang, Two-dimensional pca: a new approach to appearance-based face representation and recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 131–137.
[2] A.W.-K. Kong, D.D. Zhang, M.S. Kamel, A survey of palmprint recognition, Pattern Recognition 42 (7) (2009) 1408–1418.
[3] J.J. Koo, A.C. Evans, W.J. Gross, 3-d brain mri tissue classification on fpgas, IEEE Transactions on Image Processing 18 (12) (2009) 2735–2746.
[4] G. Shakhnarovich, T. Darrell, P. Indyk, Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing), The MIT Press, 2006.
[5] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, NY, USA, 1995.
[6] B. Scholkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.
[7] C.M. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag, Secaucus, NJ, USA, 2006.
[8] T. Joachims, Training linear svms in linear time, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, ACM, New York, NY, USA, 2006, pp. 217–226.
[9] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27.
[10] S. Rendle, Factorization machines with libfm, ACM Transactions on Intelligent Systems and Technology (TIST) 3 (3) (2012) 57.
[11] Y. Zhu, X. Wang, E. Zhong, N.N. Liu, H. Li, Q. Yang, Discovering spammers in social networks, in: AAAI, 2012.
[12] C. Hou, C. Zhang, Y. Wu, Y. Jiao, Stable local dimensionality reduction approaches, Pattern Recognition 42 (9) (2009) 2054–2066.
[13] D.L. Donoho, High-dimensional data analysis: the curses and blessings of dimensionality, in: American Mathematical Society Conference on Mathematical Challenges of the 21st Century, 2000.
[14] D. Cai, X. He, Y. Hu, J. Han, T. Huang, Learning a spatially smooth subspace for face recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Machine Learning (CVPR'07), 2007.
[15] K. Ma, X. Tang, Discrete wavelet face graph matching, in: ICIP01, 2001, pp. II: 217–220.
[16] M.A.O. Vasilescu, D. Terzopoulos, Multilinear subspace analysis of image ensembles, in: CVPR, 2003, pp. 93–99.
[17] I. Jolliffe, Principal Component Analysis, 2nd edition., Springer, New York, 2002.
[18] J. Ye, Q. Li, LDA/QR: an efficient and effective dimension reduction algorithm and its theoretical foundation, Pattern Recognition 37 (4) (2004) 851–854.
[19] X. He, P. Niyogi, Locality preserving projections, in: NIPS, 2003.
[20] F. Nie, S. Xiang, Y. Song, C. Zhang, Extracting the optimal dimensionality for local tensor discriminant analysis, Pattern Recognition 42 (1) (2009) 105–114.
[21] J. Ye, R. Janardan, Q. Li, Two-dimensional linear discriminant analysis, in: NIPS, 2004.
[22] M. Li, B. Yuan, 2d-lda: A statistical linear discriminant analysis for image matrix, Pattern Recognition Letters 26 (5) (2005) 527–532.
[23] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, H. Zhang, Multilinear discriminant analysis for face recognition, IEEE Transactions on Image Processing 16 (1) (2007) 212–220.
[24] X. He, D. Cai, P. Niyogi, Tensor subspace analysis, in: NIPS, 2005.
[25] D. Tao, X. Li, X. Wu, W. Hu, S.J. Maybank, Supervised tensor learning, Knowledge and Information Systems 13 (1) (2007) 1–42.
[26] D. Tao, X. Li, X. Wu, S.J. Maybank, Tensor rank one discriminant analysis—a convergent method for discriminative multilinear subspace selection, Neurocomputing 71 (10–12) (2008) 1866–1882.
[27] S. Chen, Z. Wang, Y. Tian, Matrix-pattern-oriented ho-kashyap classifier with regularization learning, Pattern Recognition 40 (5) (2007) 1533–1543.
[28] Z. Wang, S. Chen, J. Liu, D. Zhang, Pattern representation in feature extraction and classifier design: matrix versus vector, IEEE Transactions on Neural Networks 19 (5) (2008) 758–769.

[29] Z. Zhang, T.W.S. Chow, Maximum margin multisurface support tensor machines with application to image classification and segmentation, Expert Systems with Applications 39 (1) (2012) 849–860.
[30] C. Hou, F. Nie, D. Yi, Y. Wu, Efficient image classification via multiple rank regression, IEEE Transactions on Image Processing 22 (1) (2013) 340–352.
[31] Z. Wang, S. Chen, D. Gao, A novel multi-view learning developed from single-view patterns, Pattern Recognition 44 (10-11) (2011) 2395–2413.
[32] C.M. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[33] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, NY, USA, 2004.
[34] A.N. Langville, W.J. Stewart, The Kronecker product and stochastic automata networks, Journal of Computational and Applied Mathematics (2003) 429–447.

Chenping Hou received his B.S. and Ph.D. degrees in Applied Mathematics from National University of Defense Technology, Changsha, China, in 2004 and 2009 respectively. He is now a lecture in Department of Mathematics and System Science. He is a member of both IEEE and ACM. His current research fields include pattern recognition, machine learning and computer vision.

Feiping Nie received his B.S. degree in Computer Science from North China University of Water Conservancy and Electric Power, China, in 2000, received his MS degree in Computer Science from Lanzhou University, China, in 2003, and received his Ph.D. degree in Computer Science from Tsinghua University, China, in 2009. Currently, He is a research assistant professor at the University of Texas, Arlington, USA. His research interests include machine learning and its application fields, such as pattern recognition, data mining, computer vision, image processing and information retrieval.

Changshui Zhang received the B.S. degree in mathematics from Peking University, Beijing, China, in 1986 and the M.S. and Ph.D. degrees in control science and engineering from Tsinghua University, Beijing, China, in 1989 and 1992 respectively. In 1992, he joined the Department of Automation, Tsinghua University, and is currently a Professor. His research interests include pattern recognition, machine learning, etc. He has authored more than 200 papers. Prof. Zhang is currently an Associate Editor of the Pattern Recognition Journal.

Dongyun Yi is a professor in the Department of Mathematics and System Science at the National University of Defense Technology in Changsha, China. He has worked as a visiting researcher of the University of Warwick in 2007. His research interests include systems science, statistics and data processing.

Yi Wu is a professor in the Department of Mathematics and System Science. He earned a bachelor's and master's degree in Applied Mathematics at National University of Defense Technology in 1981 and 1988. He worked as a visiting researcher in New York State University in 1999. His research interests include applied mathematic, statistics and data processing.