Int. J. Mach. Learn. & Cyber. (2016) 7:411–426
DOI 10.1007/s13042-015-0394-x

ORIGINAL ARTICLE

Least squares recursive projection twin support vector machine for multi-class classification

Zhi-Min Yang[1] · He-Ji Wu[2] · Chun-Na Li[1] · Yuan-Hai Shao[1]

Received: 17 September 2014 / Accepted: 27 June 2015 / Published online: 10 July 2015

Abstract Multiple recursive projection twin support vector machine (MPTSVM) is a recently proposed classifier and has been proved to be outstanding in pattern recognition. However, MPTSVM is computationally expensive since it involves solving a series of quadratic programming problems. To relieve the training burden, in this paper, we propose a novel multiple least squares recursive projection twin support vector machine (MLSPTSVM) based on least squares recursive projection twin support vector machine (LSPTSVM) for multi-class classification problem. For a $K(K > 2)$ classes classification problem, MLSPTSVM aims at seeking K groups of projection axes, one for each class that separates it from all the other. Due to solving a series of linear equations, our algorithm tends to relatively simple and fast. Moreover, a recursive procure is introduced to generate multiple orthogonal projection axes for each class to enhance its performance. Experimental results on several synthetic and UCI datasets, as well as on relatively large datasets demonstrate that our MLSPTSVM has comparable classification accuracy while takes significantly less computing time compared with MPTSVM, and also obtains better performance than several other SVM related methods being used for multi-class classification problem.

Keywords Pattern recognition · Multi-class classification · Multiple recursive projection · Projection twin support vector machine · Least squares recursive projection twin support vector machine

✉ Yuan-Hai Shao
shaoyuanhai21@163.com

Zhi-Min Yang
yzm9966@126.com

He-Ji Wu
flary005@163.com

Chun-Na Li
na1013na@163.com

[1] Zhijiang College, Zhejiang University of Technology, Hangzhou 310024, People's Republic of China

[2] College of Science, Zhejiang University of Technology, Hangzhou 310023, People's Republic of China

1 Introduction

Support vector machine (SVM) [1, 2], being widely used for pattern classification and regression problems, was introduced by Vapnik and his co-workers in the early 1990s. Previous studies demonstrated the superiority of SVM [3–5]. By employing the structural risk minimization (SRM) principle [6], SVM tries to find a decision hyperplane that separates data points from two classes well by constructing two parallel support hyperplanes that the margin between them is maximized. However, SVM needs to solve a quadratic programming problem (QPP), which restricts its application to large scale problems. To address this issue, numerous approaches have been proposed [7–12].

For binary classification, some nonparallel hyperplane classifiers have attracted much attention. Mangasarian and Wild [9] proposed a generalized eigenvalue proximal support vector machine (GEPSVM) which aims at finding two nonparallel hyperplanes such that each hyperplane is closer to its own class and far from the other class as much as possible. This idea leads to solving two generalized eigenvalue problems which in turn reduces computation cost compared with SVM. Subsequently, an improved version of GEPSVM, called IGEPSVM [13], is proposed to remove the possible singularity problem of GEPSVM. In the spirit of GEPSVM, Jayadeva et. al. [10] proposed the twin support vector machine (TWSVM). Different from SVM, TWSVM solves two smaller sized QPPs rather than a single large one, which makes the learning speed of TWSVM be approximately four times faster than that of SVM. Both GEPSVM and TWSVM share the idea of nonparallel SVMs, which is in fact has been studied extensively [14–20].

Instead of finding nonparallel hyperplanes, the idea of seeking projection axes for SVMs has also been established. Ye et. al. [21] proposed a multi-weight vector projection support vector machine (MVSVM), whose purpose is to find two optimal weight vector projection directions, such that each class is centered around its own class centroid while is separated as much as possible from the other class in the projected space. Inspired by TWSVM and MVSVM, projection twin support vector machine (PTSVM) [22, 23] is proposed recently. The central thought of PTSVM is to find a projection axis one for each class, such that the within-class variance of the projected data points of its own class is minimized meanwhile projected data points of the other class scatter away as far as possible. PTSVM can be extended to find multiple orthogonal directions by a recursive procure. To accelerate the training speed of PTSVM, Shao et al. [24] adopted the idea of least squares [7, 14] and proposed a least squares projection twin support vector machine (LSPTSVM) by considering the equality constraints. An extra regularization term is introduced in the primal problem of LSPTSVM to remove the singularity problem that may appear in PTSVM.

As a natural extension of binary classification problem, multi-class classification has also drawn many attentions. Among all the methods, SVM and its variants [25–29] have been confirmed to have outstanding performance. Generally speaking, two types of strategies are widely used when SVMs are applied. One is the "decomposition-reconstruction" strategy which involves solving a series of small sized optimization problems, including the classical "one versus one" and "one versus rest" techniques [26–29]. The other one is the "all-together" strategy through solving one large scale optimization problem [25].

Being a successful multi-class classification tool, multiple recursive projection twin support vector machine (MPTSVM) [29] is a recently proposed SVM-type classifier, which is established based on binary PTSVM by utilizing the "one versus rest" technique. Though MPTSVM performs satisfactorily, it is computationally expensive since a series of QPPs are needed to be solved. For this purpose, in this paper, we extend the LSPTSVM to multi-class classification problem, and propose a multiple least squares recursive projection twin support vector machine (MLSPTSVM). Instead of solving complex QPPs in MPTSVM, our MLSPTSVM solves a series of linear equations, which leads to relatively fast training speed. For $K(K > 2)$ classes classification problem, our MLSPTSVM determines K groups of projection axes, one group for each class, such that the within-class variance of the projected data points of its own class is minimized while each projected class is far awat from the projected centers of the other classes. Specially, we apply the classical "one versus rest" multi-class classification technique to our MLSPTSVM by considering its advantage on computational efficiency. Preliminary experimental results on several real-world datasets and large datasets show the advantages of MLSPTSVM over MPTSVM and other SVM related methods for multi-class classification problem. The following are the highlights of our MLSPTSVM:

1. MLSPTSVM considers both the linear and the nonlinear models, while the binary LSPTSVM ignores the nonlinear case.
2. MLSPTSVM solves a series of linear equations, which makes it can handle the large datasets easily.
3. MLSPTSVM could generate multiple orthogonal projection directions for each class, which may notably enhance its performance.
4. The Sherman–Morrison–Woodbury (SMW) formulation and the reduced kernel technique are employed to reduce the complexity of nonlinear MLSPTSVM.

The rest of the paper is organized as follows: In Sect. 2, we provide the basic notations and give a brief review of TWSVM, PTSVM and LSPTSVM. Section 3 presents the details of MLSPTSVM under different requirement and its computational complexity. A variety of experimental results are demonstrated in Sect. 4. Finally, Sect. 5 concludes this paper.

2 Preliminaries

In this section, we consider the binary classification problem of classifying m data points in the n-dimensional real space $\mathbb{R}^n$, with data points in class 1 and class 2 are represented by the matrices $A = (\mathrm{a}_1, \ldots, \mathrm{a}_{m_1})^T \in \mathbb{R}^{m_1 \times n}$ and $B = (\mathrm{b}_1, \ldots, \mathrm{b}_{m_2})^T \in \mathbb{R}^{m_2 \times n}$ respectively, where $m = m_1 + m_2$. In the following, we will give a brief review of TSVM, PTSVM and LSPTSVM.

2.1 Twin support vector machine

Twin support vector machine (TWSVM) [10] aims at determining two nonparallel hyperplanes $x^T w^{(1)} + b^{(1)} = 0$ and $x^T w^{(2)} + b^{(2)} = 0$, such that each hyperplane is close to data points of one class and far from the data points of the other class. The two hyperplanes are obtained by solving a pair of SVM-type QPPs as the following

$$\begin{aligned}&\min_{w^{(1)},b^{(1)},\xi} \frac{1}{2}(Aw^{(1)}+e_1b^{(1)})^T(Aw^{(1)}+e_1b^{(1)})+c_1e_2^T\xi \\ &\quad \text{s.t.} -(Bw^{(1)}+e_2b^{(1)})+\xi \geq e_2, \quad \xi \geq 0\end{aligned} \tag{1}$$

and

$$\begin{aligned}&\min_{w^{(2)},b^{(2)},\eta} \frac{1}{2}(Bw^{(2)}+e_2b^{(2)})^T(Bw^{(2)}+e_2b^{(2)})+c_2e_1^T\eta \\ &\quad \text{s.t. } (Aw^{(2)}+e_1b^{(2)})+\eta \geq e_1, \quad \eta \geq 0,\end{aligned} \tag{2}$$

where $c_1, c_2 > 0$ are penalty parameters, $e_1 \in \mathbb{R}^{m_1}$ and $e_2 \in \mathbb{R}^{m_2}$ are vectors of ones, and ξ, η are vectors of nonnegative slack variables.

Through the K.K.T conditions [30], we can obtain the Wolf dual forms of problems (1) and (2) as follows

$$\begin{aligned}&\min_{\alpha} \frac{1}{2}\alpha^T P(Q^TQ)^{-1}P^T\alpha - e_2^T\alpha \\ &\text{s.t. } 0 \leq \alpha \leq c_1e_2\end{aligned} \tag{3}$$

and

$$\begin{aligned}&\min_{\gamma} \frac{1}{2}\gamma^T Q(P^TP)^{-1}Q^T\gamma - e_1^T\gamma \\ &\text{s.t. } 0 \leq \gamma \leq c_1e_1.\end{aligned} \tag{4}$$

Here, $Q=[A,e_1]$, $P=[B,e_2]$, and $\alpha=(\alpha_1,\alpha_2,\ldots,\alpha_{m_2})^T$ and $\gamma=(\gamma_1,\gamma_2,\ldots,\gamma_{m_1})^T$ are the vectors of Lagrange multipliers.

Define $u=[w^{(1)},b^{(1)}]^T$ and $v=[w^{(2)},b^{(2)}]^T$. Then the two nonparallel hyperplanes can be obtained from the solution of problems (3) and (4), which are given by

$$u = -(Q^TQ)^{-1}P^T\alpha \tag{5}$$

and

$$v = (P^TP)^{-1}Q^T\gamma \tag{6}$$

respectively. Once vectors u and v are known from (5) and (6), the two separating hyperplanes $x^Tw^{(1)}+b^{(1)}=0$ and $x^Tw^{(2)}+b^{(2)}=0$ are obtained and the training process is finished. For predicting, a new data point $\mathrm{x} \in \mathbb{R}^{\mathrm{n}}$ is assigned to class $i(i=1,2)$, depending on which of the two hyperplanes it lies closer to, i.e.,

$$label(\mathrm{x}) = \operatorname*{argmin}_{i=1,2}|\mathrm{x}^Tw^{(i)}+b^{(i)}|, \tag{7}$$

where $|\cdot|$ is the absolute value operation that computes perpendicular distance of x to each hyperplane.

It should be noted that, Q^TQ and P^TP in (5) and (6) are positive semidefinite, and hence it is possible that they may not be well defined when taking inverse. Therefore, a regularization term ϵI can be introduced to avoid the possible ill-condition of Q^TQ and P^TP, where $\epsilon > 0$ and I is the identity matrix of appropriate dimension.

2.2 Projection twin support vector machine

Different from TWSVM who finds nonparallel hyperplanes, projection twin support vector machine (PTSVM) [22] aims at finding two projection axes w_1 and w_2, one for each class, such that the within-class variance of projected data points of its own class is minimized while projected data points from other class scatter away as far as possible. This idea leads to the formulations of PTSVM as

$$\begin{aligned}&\min_{w_1} \frac{1}{2}\sum_{i=1}^{m_1}\left(w_1^T\mathrm{a}_i - w_1^T\frac{1}{m_1}\sum_{j=1}^{m_1}\mathrm{a}_j\right)^2 + c_1\sum_{k=1}^{m_2}\xi_k \\ &\text{s.t. } w_1^T\mathrm{b}_k - w_1^T\frac{1}{m_1}\sum_{j=1}^{m_1}\mathrm{a}_j + \xi_k \geq 1, \quad \xi_k \geq 0, \quad k=1,2,\ldots,m_2\end{aligned} \tag{8}$$

and

$$\begin{aligned}&\min_{w_2} \frac{1}{2}\sum_{i=1}^{m_2}\left(w_2^T\mathrm{b}_i - w_2^T\frac{1}{m_2}\sum_{j=1}^{m_2}\mathrm{b}_j\right)^2 + c_2\sum_{k=1}^{m_1}\eta_k \\ &\text{s.t. } -\left(w_2^T\mathrm{a}_k - w_2^T\frac{1}{m_2}\sum_{j=1}^{m_2}\mathrm{b}_j\right) + \eta_k \geq 1, \quad \eta_k \geq 0, \\ &\quad k=1,2,\ldots,m_1,\end{aligned} \tag{9}$$

where c_1 and c_2 are trade-off parameters, and ξ_k, η_k are nonnegative slack variables.

Before determining solutions of (8) and (9), we first define

$$S_1 = \sum_{i=1}^{m_1}\left(\mathrm{a}_i - \frac{1}{m_1}\sum_{j=1}^{m_1}\mathrm{a}_j\right)\left(\mathrm{a}_i - \frac{1}{m_1}\sum_{j=1}^{m_1}\mathrm{a}_j\right)^T \tag{10}$$

and

$$S_2 = \sum_{i=1}^{m_2}\left(\mathrm{b}_i - \frac{1}{m_2}\sum_{j=1}^{m_2}\mathrm{b}_j\right)\left(\mathrm{b}_i - \frac{1}{m_2}\sum_{j=1}^{m_2}\mathrm{b}_j\right)^T, \tag{11}$$

as the covariance matrices of the first and second class, respectively. Then problems (8) and (9) can be solved through their Wolf dual forms [30] which are given by

$$\begin{aligned}&\min_{\alpha} \frac{1}{2}\alpha^T\left(B - \frac{1}{m_1}e_2e_1^TA\right)S_1^{-1}\left(B - \frac{1}{m_1}e_2e_1^TA\right)^T\alpha - e_2^T\alpha \\ &\text{s.t. } 0 \leq \alpha \leq c_1e_2\end{aligned} \tag{12}$$

and

$$\begin{aligned}&\min_{\gamma} \frac{1}{2}\gamma^T\left(A - \frac{1}{m_2}e_1e_2^TB\right)S_2^{-1}\left(B - \frac{1}{m_2}e_1e_2^TB\right)^T\gamma - e_1^T\gamma \\ &\text{s.t. } 0 \leq \gamma \leq c_2e_1,\end{aligned} \tag{13}$$

respectively, where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{m_2})^T$ and $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_{m_1})^T$ are Lagrange multiplier vectors, $e_1 \in \mathbb{R}^{m_1}$ and $e_2 \in \mathbb{R}^{m_2}$ are vectors of ones.

After obtaining α and β, w_1 and w_2 can be expressed by

$$w_1 = S_1^{-1}\left(B - \frac{1}{m_1}e_2e_1^TA\right)^T\alpha,\qquad w_2 = S_2^{-1}\left(A - \frac{1}{m_2}e_1e_2^TB\right)^T\gamma. \tag{14}$$

For a new coming data point $\mathrm{x} \in \mathbb{R}^n$, it is assigned to class $i(i = 1, 2)$ depending on which of the two projected class centers it is closer to, i.e.,

$$label(\mathrm{x}) = \arg\min_{i=1,2}\{d_1, d_2\}, \tag{15}$$

where d_1 and d_2 represent the distances between the projection of x and the projected center of corresponding class, which are given by

$$d_1 = \left| w_1^T\mathrm{x} - w_1^T\frac{1}{m_1}\sum_{j=1}^{m_1}\mathrm{a}_j\right| \tag{16}$$

and

$$d_2 = \left| w_2^T\mathrm{x} - w_2^T\frac{1}{m_2}\sum_{j=1}^{m_2}\mathrm{b}_j\right|, \tag{17}$$

respectively.

It should be noticed that the above procedure requires the two variance matrices defined by (10) and (11) to be nonsingular. However, when there are not sufficient samples, the two variance matrices can be singular. To deal with this problem, PCA plus LDA [31, 32] technique has been employed in PTSVM. Furthermore, to enhance the performance, a recursive procure is introduced in PTSVM to obtain multiple projection axes for each class.

2.3 Least squares projection twin support vector machine

By considering equality constraints in the primal problems of PTSVM, least squares projection twin support vector machine (LSPTSVM) [24] is proposed. However, LSPTSVM is not a direct least squares version of PTSVM. In fact, LSPTSVM introduces a regularization term in the primal objective function to remove the singularity problem that may happen in PTSVM and gains better generalization ability. This leads to following optimal problems

$$\begin{aligned}&\min_{w_1}\frac{1}{2}\sum_{i=1}^{m_1}\left(w_1^T\mathrm{a}_i - w_1^T\frac{1}{m_1}\sum_{j=1}^{m_1}\mathrm{a}_j\right)^2 + \frac{c_1}{2}\sum_{k=1}^{m_2}\xi_k^2 + \frac{v_1}{2}\|w_1\|^2\\&\text{s.t. } w_1^T\mathrm{b}_k - w_1^T\frac{1}{m_1}\sum_{j=1}^{m_1}\mathrm{a}_j + \xi_k = 1,\quad k = 1, 2, \ldots, m_2\end{aligned} \tag{18}$$

and

$$\begin{aligned}&\min_{w_2}\frac{1}{2}\sum_{i=1}^{m_2}\left(w_2^T\mathrm{b}_i - w_2^T\frac{1}{m_2}\sum_{j=1}^{m_2}\mathrm{b}_j\right)^2 + \frac{c_2}{2}\sum_{k=1}^{m_1}\eta_k^2 + \frac{v_2}{2}\|w_2\|^2\\&\text{s.t. } -\left(w_2^T\mathrm{a}_k - w_2^T\frac{1}{m_2}\sum_{j=1}^{m_2}\mathrm{b}_j\right) + \eta_k = 1,\quad k = 1, 2, \ldots, m_1,\end{aligned} \tag{19}$$

where $c_1 > 0, c_2 > 0$ are trade-off parameters, $v_1 > 0, v_2 > 0$ are regularization parameters, and ξ_k, η_k are slack variables.

By substituting the equality constraint into the objective function, we can derive the optimal axes w_1 and w_2 of (18) and (19) as

$$\begin{aligned}w_1 = &\left[\frac{S_1}{c_1} + \left(-B + \frac{1}{m_1}e_2e_1^TA\right)^T\left(-B + \frac{1}{m_1}e_2e_1^TA\right) + \frac{v_1}{c_1}I\right]^{-1}\\&\times\left(B - \frac{1}{m_1}e_2e_1^TA\right)^Te_2\end{aligned} \tag{20}$$

and

$$\begin{aligned}w_2 = -&\left[\frac{S_2}{c_2} + \left(A - \frac{1}{m_2}e_1e_2^TB\right)^T\left(A - \frac{1}{m_2}e_1e_2^TB\right) + \frac{v_2}{c_2}I\right]^{-1}\\&\times\left(A - \frac{1}{m_2}e_1e_2^TB\right)^Te_1,\end{aligned} \tag{21}$$

where $e_1 \in \mathbb{R}^{m_1}$, $e_2 \in \mathbb{R}^{m_2}$ are vectors of ones, I is the identity matrix with appropriate dimension, and S_1, S_2 are defined as in (10) and (11) respectively. It is clear that different from PTSVM, LSPTSVM will not encounter the singularity problem due to the nonsingularity of the involved matrices $(\frac{S_1}{c_1} + (-B + \frac{1}{m_1}e_2e_1^TA)^T(-B + \frac{1}{m_1}e_2e_1^TA) + \frac{v_1}{c_1}I)$ and $(\frac{S_2}{c_2} + (A - \frac{1}{m_2}e_1e_2^TB)^T(A - \frac{1}{m_2}e_1e_2^TB) + \frac{v_2}{c_2}I)$. This makes LSPTSVM much more stable.

In order to further enhance the performance of LSPTSVM, multiple orthogonal directions for each class can also be obtained by using a recursive procedure. Suppose that the two groups of desired projection axes are $W_1 = \{w_1^{(t)}, t = 1, 2, \ldots, r\}$ and $W_2 = \{w_2^{(t)}, t = 1, 2, \ldots, r\}$, where $w_1^{(t)}$ and $w_2^{(t)}$ are obtained from (20) and (21) recursively, and r is the desired number of projection axes for each class. For testing, the label of a new coming data point $\mathrm{x} \in \mathbb{R}^{\mathrm{n}}$ can be similarly determined by (15), but with d_1 and d_2 are newly defined by

$$d_1 = \left|\left|W_1^T \mathrm{x} - W_1^T \frac{1}{m_1} \sum_{j=1}^{m_1} \mathrm{a}_j \right|\right| \tag{22}$$

and

$$d_2 = \left|\left|W_2^T \mathrm{x} - W_2^T \frac{1}{m_2} \sum_{j=1}^{m_2} \mathrm{b}_j \right|\right|, \tag{23}$$

respectively, where $||\cdot||$ represents the 2-norm of a vector.

3 Multi-class least squares recursive projection twin support vector machine

In this section, we consider multi-class classification problem with the dataset $T = \{(X, Y)\}$ contains $K \geq 2$ classes, where $X = \{\mathrm{x}_1, \ldots, \mathrm{x}_m\}$ consists of m data points and each data point is an n-dimensional vector, and $Y \in \mathbb{R}^m$ is the corresponding output with each element belonging to $\{1, \ldots, K\}$. We further organize data points in the i-th class as $A_i = (\mathrm{x}_{i1}, \ldots, \mathrm{x}_{im_i})^T \in \mathbb{R}^{m_i \times n}$ and define $B_i = (\overline{\mathrm{x}}_{i1}, \ldots, \overline{\mathrm{x}}_{i\overline{m}_i})^T \in \mathbb{R}^{\overline{m}_i \times n}$ as the set of the rest $K-1$ classes, where m_i is the number of data points in the i-th class and $\overline{m}_i = m - m_i$ represents the number of data points in the rest $K-1$ classes. In the following, we will present our multi-class least squares recursive projection twin support vector machine (MLSPTSVM) for the above K classes classification problem.

3.1 Linear MLSPTSVM

3.1.1 One projection axis

For K classes classification problem, linear MLSPTSVM generates K projection axes in the primal space, one for each class that separates one class from all the other classes in the same manner of binary LSPTSVM. Specifically, it requires that the projected points of one class are close to its projected center as much as possible while the distance from the projected center to the rest $K-1$ classes are far away to some extent. Denote w_i the projection axis for the i-th class, $i = 1, 2, \ldots K$. Then linear MLSPTSVM considers the following problem

$$\begin{aligned} &\min_{w_i} \frac{1}{2} \sum_{j=1}^{m_i} \left(w_i^T \mathrm{x}_{ij} - w_i^T \frac{1}{m_i} \sum_{s=1}^{m_i} \mathrm{x}_{is} \right)^2 + \frac{c_i}{2} \sum_{k=1}^{\overline{m}_i} \xi_k^2 + \frac{v_i}{2} \|w_i\|^2 \\ &\text{s.t. } w_i^T \overline{\mathrm{x}}_{ik} - w_i^T \frac{1}{m_i} \sum_{s=1}^{m_i} \mathrm{x}_{is} + \xi_k = 1, \quad k = 1, 2, \ldots, \overline{m}_i, \end{aligned} \tag{24}$$

where $c_i > 0$ is the trade-off parameter, $v_i > 0$ is the regularization parameter, and ξ_k are slack variables. We now give the geometric interpretation of problem (24). By employing the quadratic loss function, the first two terms of the objective function and the constraint are committed to minimize the empirical risk, which ensures the projected points of its own class are clustered around its projected center meanwhile the distances from the projected center to the rest classes are far away to some extents. Note that the nonnegative constraints of ξ_k are abandoned due to the usage of quadratic loss function. The last term in the objective function is a regularization term which is utilized to avoid the singularity problem of our model and reach better generalization ability, which is similar to classical SVM [1, 2] and improved TWSVM [15].

Problem (24) can be solved by the following process. We first define the within-class variance matrix S_i for the i-th class as

$$S_i = \left(A_i - \frac{1}{m_i} e_i e_i^T A_i \right)^T \left(A_i - \frac{1}{m_i} e_i e_i^T A_i \right), \tag{25}$$

where $e_i \in \mathbb{R}^{m_i}$ is the vector of ones. Substituting the constraint into the objective function, problem (24) is converted into an unconstrained problem which is given by

$$\min_{w_i} \frac{1}{2} w_i^T S_i w_i + \frac{c_i}{2} \left|\left| -B_i w_i + \frac{1}{m_i} \overline{e}_i e_i^T A_i w_i + \overline{e}_i \right|\right|^2 + \frac{v_i}{2} \|w_i\|^2, \tag{26}$$

where $\overline{e}_i \in \mathbb{R}^{\overline{m}_i}$ is the vector of ones. Set the gradient of the objective function in (26) with respect to w_i to zero, then

$$\begin{aligned} &S_i w_i + c_i \left(-B_i + \frac{1}{m_i} \overline{e}_i e_i^T A_i \right)^T \left(-B_i w_i + \frac{1}{m_i} \overline{e}_i e_i^T A_i w_i + \overline{e}_i \right) \\ &\quad + v_i w_i = 0. \end{aligned} \tag{27}$$

For simplicity, we define $H_i = (A_i - \frac{1}{m_i} e_i e_i^T A_i) \in \mathbb{R}^{m_i \times n}$ and $G_i = (B_i - \frac{1}{m_i} \overline{e}_i e_i^T A_i) \in \mathbb{R}^{\overline{m}_i \times n}$. Therefore, the optimal projection vector w_i of problem (24) can be obtained from (27) by

$$w_i = \left(\left(\frac{1}{c_i} H_i^T H_i + \frac{v_i}{c_i} I \right) + G_i^T G_i \right)^{-1} G_i^T \overline{e_i}. \tag{28}$$

Here I is the identity matrix with appropriate dimension. Owes to the extra regularization term in problem (24), the singularity issue is ruled out since the involved matrix in (28) is positive definite.

After K optimal projection axes $w_i (i = 1, \ldots, K)$ are obtained from (28), the training stage is complete. For predicting, the label of a new coming data point $\mathrm{x} \in \mathbb{R}^{\mathrm{n}}$ is determined depending on which of the class center it is closer to in the projected space:

$$label(\mathrm{x}) = \arg\min_{i=1,\ldots,K} \left| w_i^T \mathrm{x} - w_i^T \frac{1}{m_i} e_i^T A_i \right|. \tag{29}$$

The whole process above leads to the following Algorithm 1.

Algorithm 1 Linear MLSPTSVM for one projection axis.

```
Input: Dataset: $X = \{x_1, \ldots, x_m\}$;  test data: x.
Output: Projection axes: $\{w_i | i = 1, \ldots, K\}$;  the predicted label of x: $y$.
Process:
(1) Initialization: Let the class number $i = 1$.
(2) while $i \le K$.
      do
        Select the penalty parameters $c_i$ and $v_i$;
        Compute the corresponding variance matrix $S_i$ by (25);
        Determine the solution $w_i$ of (24) by(28);
        $i = i + 1$.
      end
(3) Predicting: Assign x to class $y = label(x)$ by (29).
```

3.1.2 *Multiple orthogonal projection axes*

In order to further enhance the performance of our MLSPTSVM, we can obtain multiple orthogonal projection axes for each class by a recursive procure. The recursive procure contains two steps: (i) determine projection axes $w_i (i = 1, \ldots, K)$ one for each class by carrying out Algorithm 1, and normalize w_i to have unit norm, i.e., $w_i = w_i / \|w_i\|$; (ii) generate new data points by projecting the original data points into projection subspace which is orthogonal to projection axis w_i.

Denote $W_i = \{w_i^{(j)} | j = 1, \ldots, r\}$ $(1 \le i \le K)$ as the set of multiple projection axes of the i-th class, where $w_i^{(j)}$ is the j-th projection axis and r is the desired number of projection axes of the i-th class, respectively. Then, the decision function of a new coming data $\mathrm{x} \in \mathbb{R}^n$ for multiple projection axes case is given by

$$label(\mathrm{x}) = \arg\min_{i=1,\ldots,K} \left\| W_i^T \mathrm{x} - W_i^T \frac{1}{m_i} e_i^T A_i \right\|. \tag{30}$$

Suppose that $X^{(j)}$ is the j-th projected dataset and $\mathrm{x}_l^{(j)}$ is the l-th data point in dataset $X^{(j)}$ $(j = 1, \ldots, r; l = 1, \ldots, m)$. Then, the proposed recursive MLSPTSVM algorithm works as in Algorithm 2.

Algorithm 2 Linear recursive MLSPTSVM for multiple projection axes.

```
Input: Dataset: $X = \{x_1, \ldots, x_m\}$;  test data: x;  desired projection axes number: $r$.
Output: Projection axes: $W_i = \{w_i^{(j)} | j = 1, \ldots, r\}$ $(i = 1, \ldots, K)$; the predicted label of x: $y$.
Process:
(1)Initialization: Let the training set $X^{(1)} = X = \{x_l^{(1)} = x_l, l = 1, \ldots, m\}$, the class number
   $i = 1$, and the axis number $j = 1$;
(2)Determine $w_i^{(j)}$ on dataset $X^{(j)}$ by implementing Algorithm 1, and normalize $w_i^{(j)}$ to have
   unit norm, i.e., $w_i^{(j)} = w_i^{(j)} / \|w_i^{(j)}\|$;
(3)for $j = 2, \ldots, r$
     do
     $X^{(j)} = \{x_l^{(j)} | x_l^{(j)} = x_l^{(j-1)} - w_i^{(j-1)} w_i^{(j-1)^T} x_l^{(j-1)}, l = 1, \ldots, m\}$;
     Implement (2);
   end
(4) If $i < K$, let $i = i + 1$, $j = 1$ and go back to step (2); otherwise, go to step (5);
(5) Predicting: Assign x to class $y = label(x)$ by (30).
```

We show that the multiple projection axes obtained for each class by Algorithm 2 is actually orthogonal to each other.

Theorem 1 *By implementing Algorithm 2, the resulting W_i is an orthonormal set for each $i = 1, \ldots, K$.*

Proof We here take the similar strategy in [22, 33, 34]. Since each $w_i^{(j)}$ is a unit vector by performing Algorithm2, we need only to prove that w_i^j is orthogonal to $w_i^{(j-k)}$ for all $k = 1, 2, \ldots, j-1$. According to the definition of G_i and $\overline{e_i}$ in Sect. 3.1.1, $G_i^T \overline{e_i}$ is a linear combination of the row vectors of matrices A_i and B_i, i.e. the input samples. By observing (28), this implies that in the j-th iteration, each projection axis $w_i^{(j)}$ is a linear combination of the input samples $\mathrm{x}_l^{(j)}$, where $l = 1, \ldots, m$. For the newly obtained projected data in the j-th iteration (Step (3) in Algorithm 2), by multiplying $w_i^{(j-1)^T}$, we have

$$w_i^{(j-1)^T} \mathrm{x}_l^{(j)} = w_i^{(j-1)^T} \mathrm{x}_l^{(j-1)} - w_i^{(j-1)^T} w_i^{(j-1)} w_i^{(j-1)^T} \mathrm{x}_l^{(j-1)} = 0.$$

This means $w_i^{(j-1)}$ is orthogonal to the input samples $\mathrm{x}_l^{(j)}$, which in turn implies that $w_i^{(j)}$ is orthogonal to $w_i^{(j-1)}$.

In the same way, we can justify that $w_i^{(j)}$ is orthogonal to $w_i^{(j-k)}$ for all $k = 2, \ldots, j-1$. Therefore, W_i is an orthonormal set for each $i = 1, \ldots, K$ and the proof is completed.

From Theorem 1, we see that each W_i spans an orthogonal subspace such that the discriminative information for classification are contained as much as possible.

3.2 Nonlinear MLSPTSVM

In this subsection, we extend the linear MLSPTSVM to the nonlinear case, which is ignored in LSPTSVM [24]. Consider the nonlinear kernel $\mathcal{K}$, and let $C = [A_1, \ldots, A_K]^T$. Then the within-class variance of the i-th class in the kernel space can be written as

$$\begin{aligned}(S_i)^\phi = &\left(\mathcal{K}(A_i, C^T) - \frac{1}{m_i} e_i e_i^T \mathcal{K}(A_i, C^T) \right)^T \\ &\times \left(\mathcal{K}(A_i, C^T) - \frac{1}{m_i} e_i e_i^T \mathcal{K}(A_i, C^T) \right),\end{aligned} \tag{31}$$

and nonlinear MLSPTSVM leads to the following unconstraint problem

$$\begin{aligned}\min_{w_i} \quad & \frac{1}{2} w_i^T (S_i)^\phi w_i \\ & + \frac{c_i}{2} \| - \mathcal{K}(B_i, C^T) w_i + \frac{1}{m_i} \overline{e_i} e_i^T \mathcal{K}(A_i, C^T) w_i + \overline{e_i} \|^2 + \frac{v_i}{2} \|w_i\|^2,\end{aligned} \tag{32}$$

where $c_i > 0$ is the trade-off parameter, $v_i > 0$ is the regularization parameter, and $e_i \in \mathbb{R}^{m_i}$, $\overline{e_i} \in \mathbb{R}^{\overline{m_i}}$ are vectors of ones. The optimal solution of problem (32) can be determined by

$$w_i = \left(\left(\frac{1}{c_i}\overline{H}_i^T\overline{H}_i + \frac{v_i}{c_i}I\right) + \overline{G}_i^T\overline{G}_i\right)^{-1}\overline{G}_i^T\overline{e_i}, \tag{33}$$

where $\overline{H}_i = \mathcal{K}(A_i, C^T) - \frac{1}{m_i}e_ie_i^T\mathcal{K}(A_i, C^T)$ and $\overline{G}_i = \mathcal{K}(B_i, C^T) - \frac{1}{m_i}\overline{e_i}e_i^T\mathcal{K}(A_i, C^T)$.

After the K optimal projection axes w_i $(i = 1, \ldots, K)$ are obtained, the label of a new coming data point x is classified in the same way as the linear case. To get multiple projection axes, the similar procedure can be taken as in Algorithm 2, which will be omitted here.

3.3 Computational analysis

We analyze the computation complexity of our MLSPTSVM in this subsection. Suppose r is the desired number of projection axes for each class. From (28), we see that linear MLSPTSVM solves K classes classification problem mainly by giving K matrix inverses of size $n \times n$, where $n \ll m$. Thus the time complexity of linear MLSPTSVM is about $O(Krn^3)$. Similarly, by observing (33), the nonlinear MLSPTSVM requires K matrix inverses of order $m \times m$, and it takes $O(Krm^3)$ time.

In order to reduce the time complexity when calculating matrix inverses for nonlinear MLSPTSVM, we resort to the following Sherman–Morrison–Woodbury (SMW) formula [30]

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^TA^{-1}U)^{-1}V^TA^{-1}. \tag{34}$$

Specifically, by (34), we can rewrite (33) as

$$w_i = (Y_i - Y_i\overline{G}_i^T(I + \overline{G}_iY_i\overline{G}_i^T)^{-1}\overline{G}_iY_i)\overline{G}_i^T\overline{e_i}, \tag{35}$$

where $Y_i = (\frac{1}{c_i}\overline{H}_i^T\overline{H}_i + \frac{v_i}{c_i}I)^{-1}$ that can be further rewritten by (34) as

$$Y_i = \frac{c_i}{v_i}\left(I - \frac{1}{v_i}\overline{H}_i^T(I + \frac{1}{v_i}\overline{H}_i\overline{H}_i^T)^{-1}\overline{H}_i\right). \tag{36}$$

As we can observe from (35) and (36), the calculation of matrix inverse in (33) is converted into two smaller sized $m \times m$ matrix inverses, that is, $(I + \frac{1}{v_i}\overline{H}_i\overline{H}_i^T)^{-1} \in \mathbb{R}^{m_i \times m_i}$ and $(I + \overline{G}_iY_i\overline{G}_i^T)^{-1} \in \mathbb{R}^{\overline{m}_i \times \overline{m}_i}$. Therefore, the computation complexity of nonlinear MLSPTSVM can be reduced to $O(Krd^3)$, where $d = max\{m_i, \overline{m}_i | i = 1, \ldots, K\}$.

Furthermore, if the number of data points m in dataset T is very large, then the rectangular kernel technique [8, 35] can be applied to reduce the dimensionality of nonlinear MLSPTSVM. Specifically, we can reduce $\mathcal{K}(A_i, C^T)$ of size $m_i \times m$ and $\mathcal{K}(B_i, C^T)$ of size $\overline{m}_i \times m$ to much smaller sizes $m_i \times \widetilde{m}$ and $\overline{m}_i \times \widetilde{m}$, respectively. Here $\widetilde{m}$ is as small as 1–10 % of m and $\overline{C}$ is an $\widetilde{m} \times n$ random submatrix of C. Thus the complexity of nonlinear MLSPTSVM can be largely reduced. The rectangular kernel technique not only makes large scale problem tractable, but also leads to improved generalization performance by avoiding data overfitting [8].

4 Experimental results

To demonstrate the classification ability of MLSPTSVM, we perform our MLSPTSVM, the recently proposed MPTSVM [29], together with other four state-of-the-art binary classification methods, including SVM [1, 2], GEPSVM [9], TWSVM [10], and LSTSVM [14] on artificial datasets, publicly available UCI datasets [36] and some large datasets [37]. Note that the SVM, GEPSVM, TWSVM and LSTSVM are applied here to multi-class classification problem by using one vs rest technique. To clarify the fact that these binary classifiers are used in the multi-classification context, we re-term them as MSVM, MGEPSVM, MTWSVM, and MLSTSVM respectively, where the first letter "M" represents "multiple". All the methods are implemented in MATLAB 2013a environment on a PC with Intel i5 processor (2.67 GHz), 2 GB RAM. MSVM is implemented by LIBSVM [38] due to its fast training speed. The dual QPPs arising in MSVM, MTWSVM and MPTSVM are solved using Mosek optimization toolbox, the eigenvalue problems in MGEPSVM are solved by a function 'eig', and the matrix inverse problems in several methods including MLSPTSVM are solved by calling operation '\'. For parameter selection, all the parameters except for the number of projection axes in MPTSVM and MLSPTSVM, are selected from 2^{-8} to 2^8 by employing the standard tenfold cross-validation technique [39]. Experiments are repeated five times on each dataset and the corresponding results are recorded, including the means and standard deviations of test accuracies, computing time which are obtained under the best parameters, and p values which are calculated by performing paired t-test in 5 % significance level. Specially, in order to compare the performances of various methods intuitively, we mark the highest accuracy on each dataset in bold.

4.1 Artificial examples

We first conduct experiments on two artificial examples to evaluate the performance of our MLSPTSVM in comparison to MPTSVM. The first dataset is a three-dimensional Xor dataset that consists of 153 samples with three classes of the same size, as shown in Fig. 1. This three-dimensional Xor dataset is obtained by randomly perturbing points around three intersecting lines. The second dataset is a cross plane dataset containing three classes, with 600 samples are generated in a two-dimensional plane, as shown in Fig. 2.

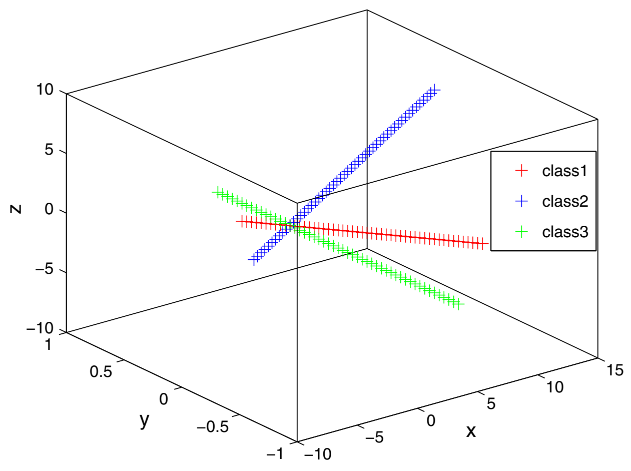


Fig. 1 A three-dimensional Xor dataset with three classes

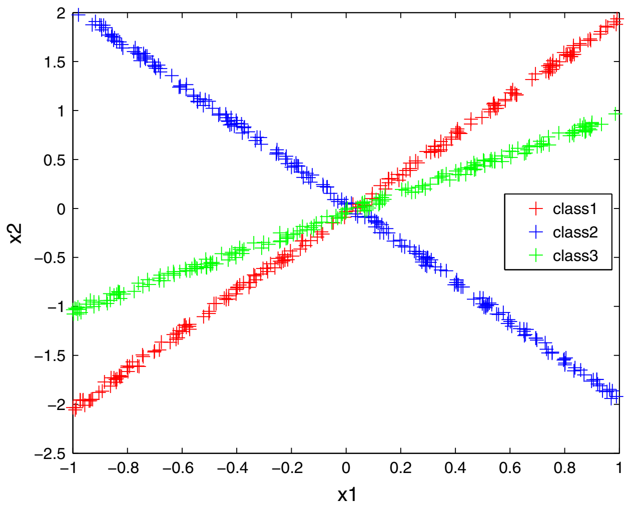


Fig. 2 A two-dimensional crossplane dataset with three classes

For the experiment on the first dataset, the number of projection axes for each class is set to 1 for both MLSPTSVM and MPTSVM. Fig. 3a, b show the classification results by plotting the distance distributions of MPTSVM and MLSPTSVM. Specifically, d_i $(i = 1, 2, 3)$ in Fig. 3 are the distances between the projected points and the i-th projected center. MLSPTSVM classifies the Xor dataset well with just two misclassified points, which is the same for MPTSVM.

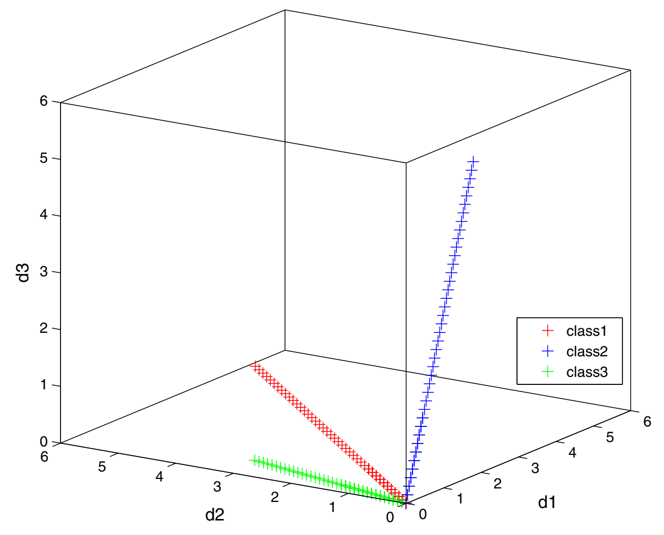
For the second artificial dataset, Fig. 4a, b depict the three projection directions obtained by MPTSVM and MLSPTSVM, respectively. From Fig. 4, we see that the resulting directions for these two methods are very similar. To make the comparison clearer, the specific performances of MPTSVM and MLSPTSVM on the two datasets are reported in Table 1. As we can observe in Table 1, MLSPTSVM has comparable classification accuracy to that of MPTSVM but with considerably less computing time. Furthermore, by observing the p values, we see that 0.775 and 0.799 are much greater than 0.05, which implies that the performance of these two methods are essentially similar.

4.2 UCI datasets

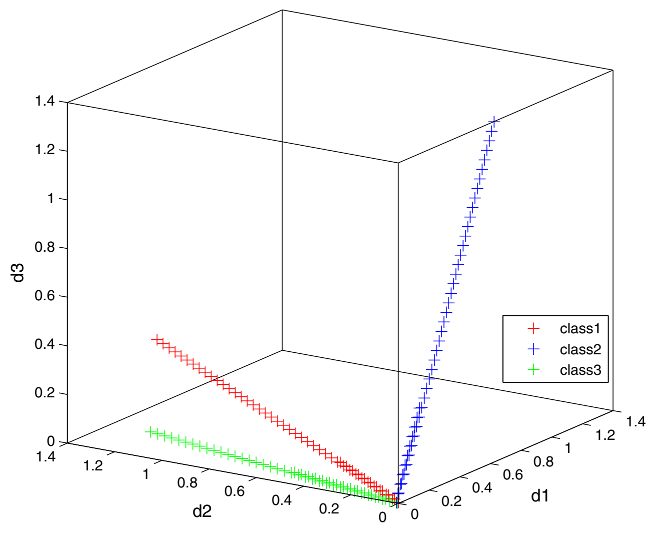
In this subsection, we further experiment on six UCI benchmark datasets [36], whose details are listed in Table 2. The experimental results for linear and nonlinear cases of our MLSPTSVM, as well as the other five methods, i.e., MSVM, MGEPSVM, MTWSVM, MLSTSVM and MPTSVM are summarized in Tables 3 4, respectively. For nonlinear classifiers, Gaussian kernel is selected for all the methods. The SMW technique [30] is employed to simplify the calculation of matrix inverses for our nonlinear MLSPTSVM. The optimal numbers of projection axes that are selected from 1 to the dimension of each dataset are listed in the brackets for MPTSVM and MLSPTSVM.

For the linear case, from Table 3, we observe that MLSPTSVM and MPTSVM can obtain the highest accuracies on most of the datasets among all the methods, which indicate that they have better generalization capability than the others. Furthermore, we see that MLSPTSVM has comparable accuracies to those of MPTSVM since most of p values between them are larger than 0.05. On the other side, MLSPTSVM takes significantly less computing time than MPTSVM as one can see in Table 3. For example, MLSPTSVM takes 0.017(s) on Glass dataset while MPTSVM needs almost 0.431(s). However, MLSPTSVM is a bit slower than MGEPSVM and MLSTSVM, which mainly because multiple projection axes are needed on most datasets for MLSPTSVM.

For the nonlinear case, the employed Gaussian kernel is defined by $\mathcal{K}(x, z) = \exp\left(-\frac{\|x-z\|^2}{2p^2}\right)$, where p is the kernel parameter. From Table 4, we can see that MLSPTSVM and MPTSVM outperform the other methods in terms of classification accuracy, while these two methods achieve comparable performance by observing both the accuracies and p values. For example, MPTSVM and MLSPTSVM obtain accuracies 97.47 and 98.01 % on Pathbased dataset, while the corresponding p value is 0.084. With regard to time consumption, from Table 4, it can be seen that MLSPTSVM takes almost the fewest computing time compared with its competitors. Specially, MLSPTSVM takes significantly less time than MPTSVM as in the linear case. Furthermore, only one projection axis is enough for MLSPTSVM to get the highest accuracy on three datasets, i.e., Seeds, Wine and Pathbased, while the corresponding

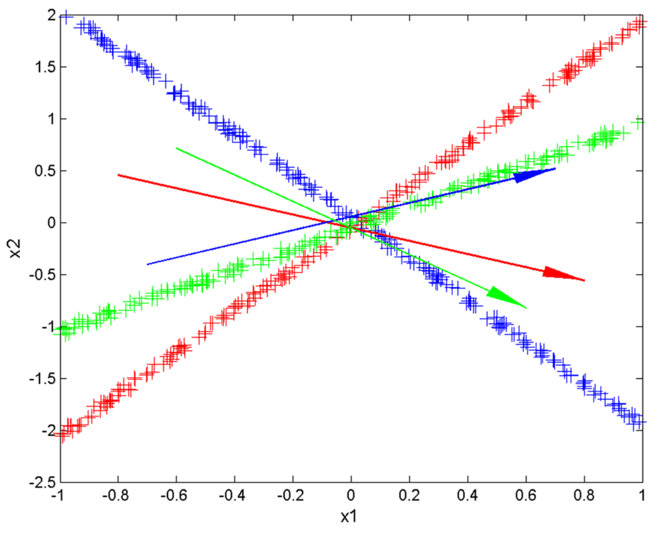


(a) MPTSVM

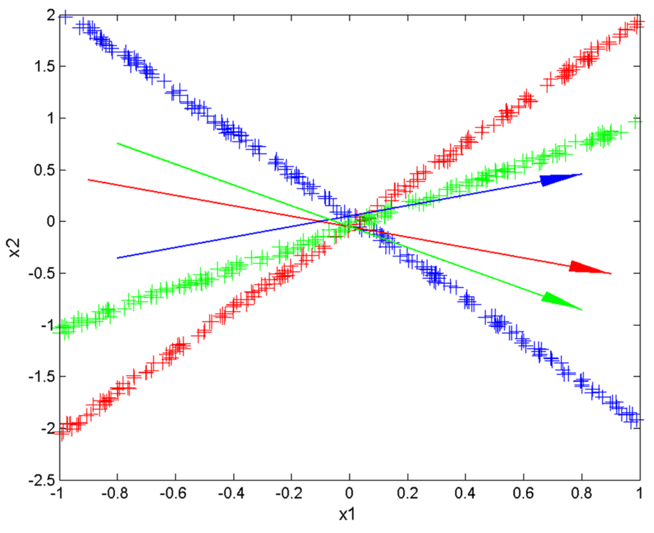


(b) MLSPTSVM

Fig. 3 Distance distribution of MPTSVM and MLSPTSVM on Xor dataset



(a) MPTSVM



(b) MLSPTSVM

Fig. 4 Projection directions of MPTSVM and MLSPTSVM on crossplane dataset

computing times are very competitive. This demonstrates the superiority of our MLSPTSVM.

In summary, from Tables 3 and 4, we conclude that MLSPTSVM and MPTSVM outperform other methods in both linear and nonlinear cases, and there is no statistical difference in classification accuracy between these two methods since most of corresponding *p* values are larger than 0.05. However, it is evident that MLSPTSVM takes considerable less computing time compared with MPTSVM. In addition, the optimal number of projection axes varies from different datasets for both MPTSVM and our MLSPTSVM.

4.3 Large datasets

As one observes in Sect. 3, our MLSPTSVM tends to extremely fast since it only needs to solve a series of linear equations. In this subsection, we argue the above viewpoint by exhibiting the ability of our MLSPTSVM on dealing with large scale datasets. Note that MSVM, MTWSVM and MPTSVM involve solving complex QPPs with high computational complexity, which makes them fail on large datasets. Therefore, we only present the experimental results of our MLSPTSVM, MGEPSVM and MLSTSVM. Eight large datasets [37] are used for comparison, whose

Table 1 Performance comparison of MPTSVM and MLSPTSVM on artificial datasets

Datasets	MPTSVM Acc ± std (%) Time (s) *p* value	MLSPTSVM Acc ± std(%) Time (s) *p* value
Xor	98.81 ± 0.20	**98.83 ± 0.06**
	0.007	0.002
	0.775	–
Crosspalne	**97.58 ± 0.31**	97.53 ± 0.16
	0.038	0.004
	0.799	–

Table 2 Details of the benchmark UCI datasets

Dataset	Points	Features	Clusters
(a) Glass	214	9	6
(b) Seeds	210	7	3
(c) Dermatology	366	34	6
(d) Wine	178	13	3
(e) Pathbased	300	2	3
(f) Zoo	101	16	7

details are summarized in Table 5. Here, the Mnist dataset is generated by randomly selecting 30 % samples in each class of the original dataset, and the other datasets are combined by the training set and testing set. Since our linear MLSPTSVM requires to determine the inversion of matrices whose orders are input space dimension, we perform dimensionality reduction step before classifying on high-dimensional datasets. Therefore, we first employ LDA [31] on USPS, Reuster300 and Mnist datasets for dimensionality reduction. Furthermore, the number of projection axes are selected from 1 to 8 for the convenience of further analyzing in Sect. 4.4.

For the linear case, we report the experimental results of the three involved methods on the above large datasets in Table 6. Table 6 demonstrate that our MLSPTSVM outperforms MLSTSVM and MGEPSVM on most datasets in terms of classification accuracy, and the corresponding *p* values between them are all much smaller than 0.05, which indicates the effectiveness of our MLSPTSVM over MLSTSVM and MGEPSVM. For example, for Shuttle dataset, MGEPSVM and MLSTSVM gain 67.77 and 90.12 % accuracy respectively while MLSPTSVM can reach to 91.19 %. Meanwhile, the two corresponding *p* values are all close to 0. It can also be observed that our MLSPTSVM is a bit slower than MGEPSVM and MLSTSVM, which is mainly because that the optimal number of projection axes is needed to be searched. However, this also the source of great performance of MLSPTSVM. In conclusion, the results in Table 6 demonstrate the feasibility of our linear MLSPTSVM on large datasets.

For the nonlinear case, four datasets in Table 5 are considered, i.e., Page-blocks, Satimage, Pendigits and Mnist, and the Gaussian kernel is used in nonlinear MLSPTSVM. We employ the rectangular kernel technique [8, 35] to reduce the dimensionality and select 1 % of training samples to perform the kernel transformation. The corresponding experimental results of nonlinear MGEPSVM, MLSTSVM and MLSPTSVM on these four large datasets are reported in Table 7. From Table 7, we find that nonlinear MLSPTSVM also handle large datasets well while with acceptable computational time. For example, for Pendigis dataset, nonlinear MLSPTSVM gains as high as 98.42 % accuracy compared with 90.55 % for linear MLSPTSVM, but with 1.56(s) computational time. The phenomena further confirms the ability of our MLSPTSVM to deal with large scale datasets. Besides, nonlinear MLSPTSVM can obtain comparable classification accuracy with MLSTSVM, while these two methods outperform MGEPSVM to a large extent.

4.4 Parameters analysis

Linear MLSPTSVM contains two sets of penalty parameters c_i and v_i, $i = 1, 2, \ldots, K$, which are needed to be selected independently. Moreover, the number of projection axes is also considered as a parameter. In this subsection, we analyze the influence of these parameters to the performance of our MLSPTSVM on four large datasets, i.e., Page-blocks, USPS, Pendigits and Shuttle, respectively. For convenience, we use the same c_i and v_i for each $i = 1, 2, \ldots, K$, that is, $c_i = c$ and $v_i = v$ for some c and v.

We first consider the effect of parameters c and v on the test accuracy, which is shown in Fig. 5. Here, the grid search method is employed, and the corresponding accuracy is obtained under the optimal number of projection axes, while parameters c and v are changing within the set $\{2^{-8}, \ldots, 2^{8}\}$. From Fig. 5, we observe that the accuracy varies greatly along with the change of parameters c and v, which implies that c and v have a great impact on the classification accuracy. Thus, to achieve better performance, it is necessary to select the suitable parameters c and v.

We next depict the relationship between the number of projection axes and classification accuracy in Fig. 6. Here, the number of projection axes is selected from 1 to 8. As we can see from Fig. 6, multiple orthogonal projection axes are required for the four datasets in order to reach higher accuracy, and for different datasets, the optimal number of projection axes varies. For example, USPS dataset needs six projection axes while Shuttle dataset only

Table 3 Performance comparison for various methods using the linear kernel

Dataset	MSVM Acc ± std (%) Time (s) *p* value	MGEPSVM Acc ± std (%) Time (s) *p* value	MTWSVM Acc ± std (%) Time (s) *p* value	MLSTSVM Acc ± std (%) Time (s) *p* value	MPTSVM(r) Acc ± std (%) Time (s) *p* value	MLSPTSVM(r) Acc ± std (%) Time (s) *p* value
(a)	58.92 ± 1.26	42.69 ± 1.72	60.34 ± 1.41	59.77 ± 1.28	**63.28 ± 1.37**(1)	61.41 ± 1.19(7)
	0.033	0.003	0.045	0.005	0.431	0.017
	0.069	0.000	0.159	0.004	0.080	–
(b)	95.47 ± 0.27	89.29 ± 1.14	95.26 ± 0.45	93.77 ± 0.32	96.91 ± 0.32(2)	**97.42 ± 0.23**(1)
	0.030	0.002	0.022	0.002	0.362	0.003
	0.001	0.000	0.001	0.000	0.050	–
(c)	96.67 ± 0.29	84.27 ± 0.59	94.43 ± 0.28	97.31 ± 0.30	97.86 ± 0.49(1)	**97.90 ± 0.22**(3)
	0.077	0.008	0.124	0.008	0.370	0.020
	0.002	0.000	0.000	0.034	0.853	–
(d)	98.38 ± 0.61	84.07 ± 0.77	98.39 ± 0.38	98.87 ± 0.62	98.91 ± 0.47(4)	**98.92 ± 0.58**(2)
	0.011	0.002	0.032	0.002	0.148	0.005
	0.014	0.000	0.168	0.453	0.937	–
(e)	65.69 ± 1.31	63.54 ± 0.25	63.28 ± 0.36	**65.90 ± 0.99**	62.73 ± 0.40(2)	62.43 ± 0.96(2)
	0.024	0.002	0.014	0.003	0.236	0.005
	0.001	0.082	0.225	0.014	0.558	–
(f)	92.51 ± 1.13	87.48 ± 1.78	94.29 ± 0.46	94.65 ± 1.15	**97.05 ± 0.07**(6)	95.18 ± 1.25(5)
	0.012	0.005	0.011	0.004	0.438	0.009
	0.049	0.001	0.168	0.567	0.032	–

Table 4 Performance comparison for various methods using the nonlinear kernel

Dataset	MSVM Acc ± std (%) Time (s) *p* value	MGEPSVM Acc ± std (%) Time (s) *p* value	MTWSVM Acc ± std (%) Time (s) *p* value	MLSTSVM Acc ± std (%) Time (s) *p* value	MPTSVM(r) Acc ± std (%) Time (s) *p* value	MLSPTSVM(r) Acc ± std (%) Time (s) *p* value
(a)	69.77 ± 1.14	60.07 ± 2.21	70.86 ± 1.70	69.65 ± 0.50	**72.54 ± 0.50**(1)	72.45 ± 0.49(3)
	0.072	0.519	0.083	0.050	0.607	0.084
	0.006	0.000	0.107	0.000	0.642	–
(b)	95.34 ± 0.58	91.95 ± 1.55	95.48 ± 0.42	96.32 ± 0.52	94.87 ± 0.29(1)	**96.40 ± 0.38**(1)
	0.018	0.200	0.028	0.022	0.326	0.019
	0.035	0.003	0.044	0.813	0.002	–
(c)	97.82 ± 0.34	74.06 ± 0.78	96.01 ± 0.25	96.40 ± 0.25	97.47 ± 0.21(2)	**98.01 ± 0.40**(2)
	0.084	2.163	0.260	0.202	2.954	0.274
	0.570	0.000	0.000	0.003	0.084	–
(d)	98.42 ± 0.56	88.44 ± 1.67	98.56 ± 0.37	99.34 ± 0.20	99.06 ± 0.11(3)	**99.58 ± 0.24**(1)
	0.013	0.132	0.029	0.017	0.321	0.015
	0.014	0.000	0.002	0.045	0.005	–
(e)	98.71 ± 0.64	90.76 ± 0.91	99.07 ± 0.19	99.52 ± 0.25	99.58 ± 0.14(2)	**99.70 ± 0.22**(1)
	0.053	0.548	0.061	0.061	0.487	0.047
	0.024	0.000	0.013	0.339	0.233	–
(f)	94.26 ± 1.96	70.74 ± 2.43	96.05 ± 0.45	95.50 ± 1.46	**96.26 ± 1.75**(1)	95.41 ± 0.83(2)
	0.017	0.063	0.025	0.018	0.285	0.025
	0.198	0.000	0.227	0.919	0.341	–

Table 5 Details of the large datasets

Dataset	Points	Features	Clusters
(a) Page-blocks	5473	10	5
(b) Satimage	6435	36	6
(c) USPS	9298	9	10
(d) Reuster300	9555	9	10
(e) Pendigits	10,992	16	10
(f) Mnist	15,000	9	10
(g) Letter	20,000	16	26
(h) Shuttle	58,000	9	7

Table 6 Performance comparison on large datasets when using the linear kernel

Dataset	MGEPSVM Acc (%) ± std Time (s) *p* value	MLSTSVM Acc (%) ± std Time (s) *p* value	MLSPTSVM Acc (%) ± std Time (s) *p* value
(a)	85.63 ± 0.13	**94.30** ± **0.08**	93.67 ± 0.11(5)
	0.025	0.037	0.047
	0.000	0.000	–
(b)	56.79 ± 0.14	78.34 ± 0.08	**78.68** ± **0.06**(8)
	0.076	0.112	0.637
	0.000	0.001	–
(c)	38.86 ± 0.16	90.58 ± 0.08	**91.16** ± **0.09**(6)
	0.106	0.147	0.240
	0.000	0.000	–
(d)	31.51 ± 0.47	56.78 ± 0.03	**60.55** ± **0.19**(8)
	0.095	0.134	0.374
	0.000	0.000	–
(e)	64.18 ± 0.32	88.53 ± 0.06	**90.55** ± **0.10**(6)
	0.118	0.165	0.405
	0.000	0.000	–
(f)	39.79 ± 0.14	**87.54** ± **0.02**	86.78 ± 0.09(8)
	0.181	0.278	0.605
	0.000	0.000	–
(g)	28.68 ± 0.26	**57.76** ± **0.07**	53.86 ± 0.12(5)
	0.534	0.799	1.868
	0.000	0.000	–
(h)	67.77 ± 0.06	90.12 ± 0.01	**97.05** ± **0.04**(2)
	0.371	0.516	0.665
	0.000	0.000	–

needs two. However, it should be noted that sometimes one projection axis is enough for MLSPTSVM, as we can observe in Tables 3 and 4. This shows that on one hand, multiple orthogonal projection axes may necessary for some datasets; on the other hand, for some datasets, redundant projection axes may bring confused classification information [40]. Moreover, by observing Fig. 6, we

Table 7 Performance comparison on large datasets when using the nonlinear kernel

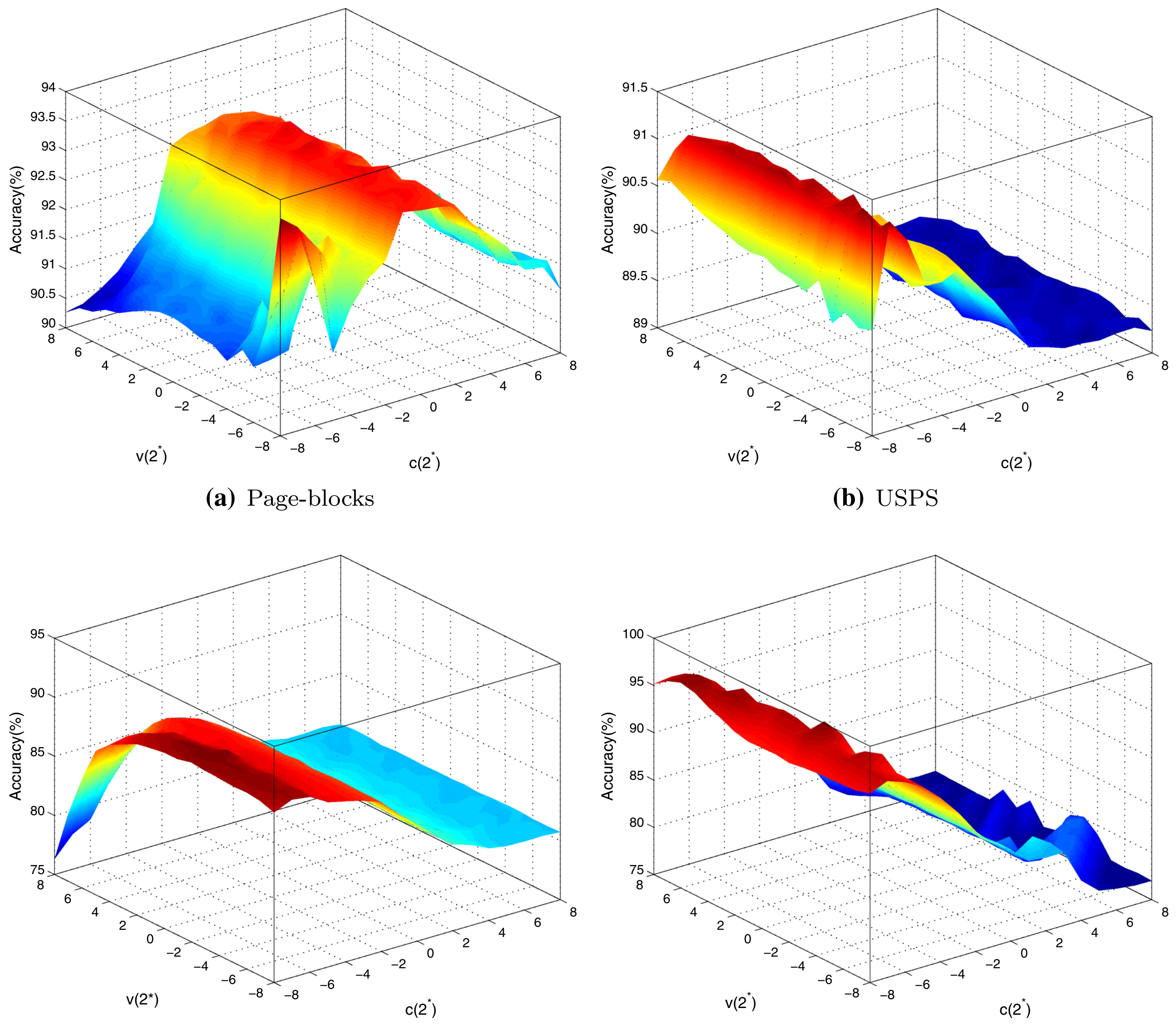
Dataset	MGEPSVM Acc (%) ± std Time (s) *p* value	MLSTSVM Acc (%) ± std Time (s) *p* value	MLSPTSVM Acc (%) ± std Time (s) *p* value
(a)	68.18 ± 7.82	**95.89** ± **0.09**	94.54 ± 0.09(1)
	0.11	0.14	0.16
	0.002	0.000	–
(b)	71.17 ± 1.18	**87.69** ± **0.17**	84.86 ± 0.29(1)
	0.243	0.293	0.317
	0.000	0.000	–
(e)	76.18 ± 0.98	98.39 ± 0.11	**98.42** ± **0.07**(1)
	1.002	1.395	1.560
	0.000	0.652	–
(f)	87.34 ± 0.11	90.23 ± 0.09	**90.81** ± **0.10**(1)
	1.625	2.612	3.050
	0.000	0.000	–

see that with the increase of the number of projection axes, the accuracy increases until it is up to a maximum, and then decreases in general, although there may be some specifications as in Fig. 6a. In summary, a proper number of projection axes can improve the performance of MLSPTSVM to a much extent.

For nonlinear MLSPTSVM, the kernel parameter p needs to be considered. We next show the relationship between kernel parameter p and classification accuracy on four UCI datasets in Table 2, i.e., Seeds, Dermatology, Wine, and Pathbased. We draw the parameter-accuracy curves in Fig. 7 on these four datasets with parameter p belonging to $\{2^{-8}, \ldots, 2^{8}\}$. Here, each accuracy is obtained under the optimal parameters c, v and the number of projection axes. From Fig. 7, we can see that kernel parameter p has a great influence on classification accuracy. For example, for dataset Wine, the lowest accuracy is 38.12 % while the highest one can reach to 100 %. Thus, a suitable kernel parameter is crucial for nonlinear MLSPTSVM to achieve better performance.

5 Conclusions

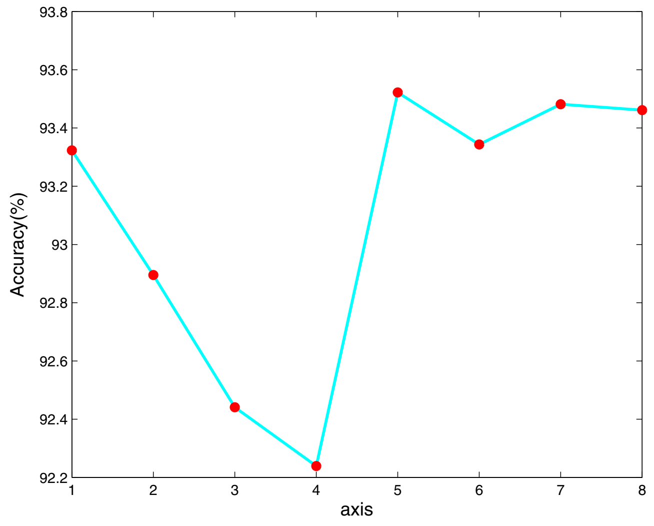
In this paper, a novel multiple least squares recursive projection twin support vector machine for multi-class classification is proposed, termed as MLSPTSVM. For K classes classification problem, our MLSPTSVM solves K groups of primal problems directly by solving a series of linear equations, which leads to its fast training speed. A recursive procure is further introduced to MLSPTSVM to generate multiple projection directions. Preliminary



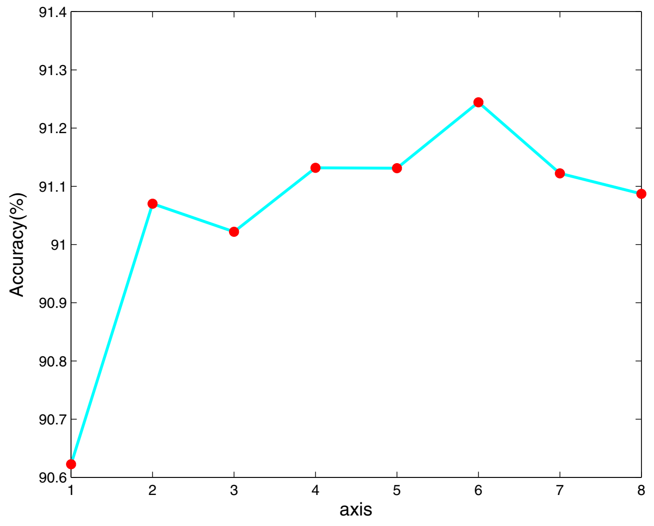
(a) Page-blocks **(b)** USPS

(c) Pendigtis **(d)** Shuttle

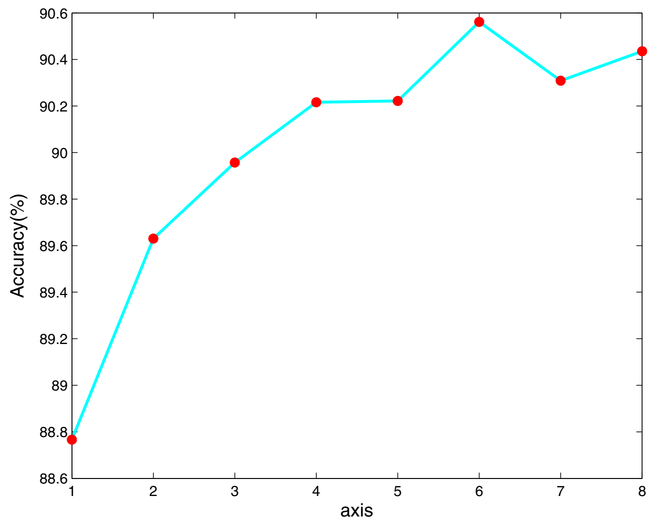
Fig. 5 Relationship between parameters *c*, *v* and classification accuracy



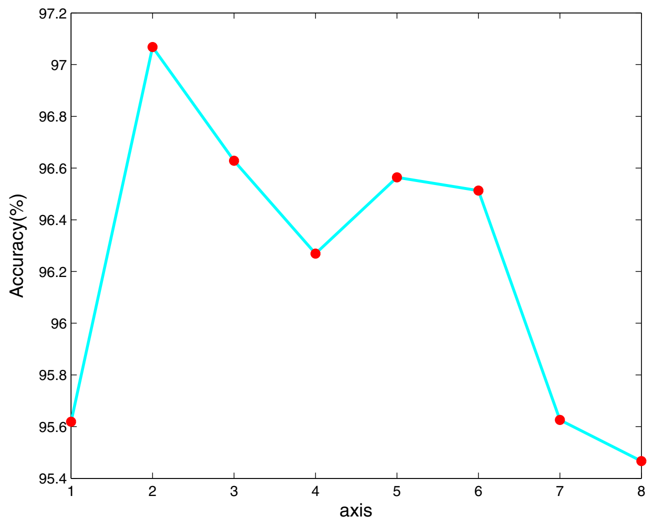
(a) Page-blocks



(b) USPS



(c) Pendigits



(d) Shuttle

Fig. 6 Relationship between the numbers of projection axes and classification accuracy

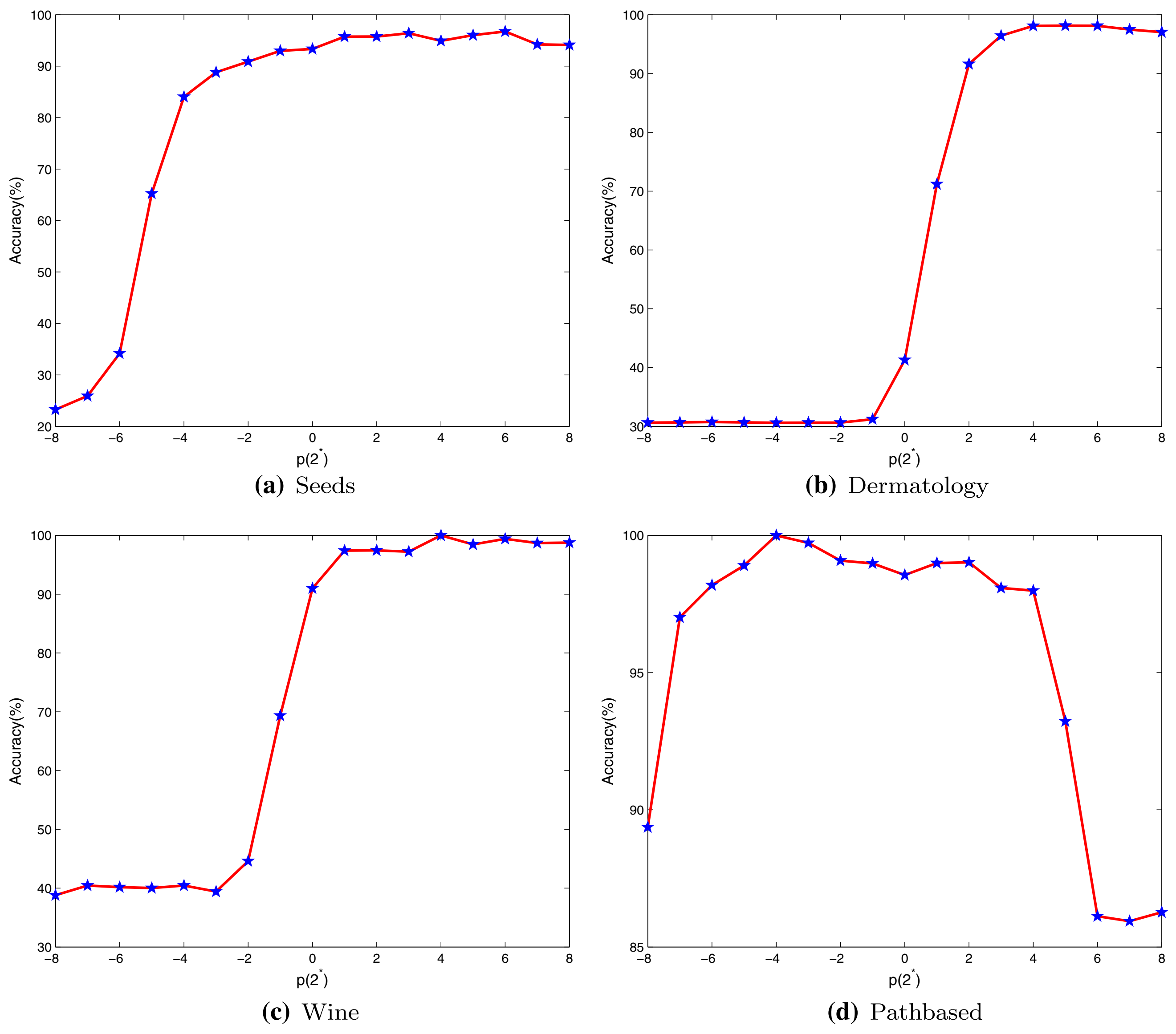


Fig. 7 Relationship between the kernel parameter and classification accuracy

experimental results show that our MLSPTSVM has comparable classification accuracy with MPTSVM but with dramatically less computing time. Besides, experiments on some large datasets further demonstrate the effectiveness of our MLSPTSVM. For practical convenience, we upload our MLSPTSVM MATLAB code upon http://www.optimal-group.org/Resource/MLSPTSVM.html. As we know, extracting features is crucial for classification, especially when faced with high-dimensional data. Thus, exploring effective feature selection/extraction methods to improve the performance of MLSPTSVM will be one of our future works.

Acknowledgments The authors would like to thank the editors and reviewers for their valuable comments and helpful suggestions, which improved the quality of this paper. This work is supported by the National Natural Science Foundation of China (Nos. 11201426, 11371365, 11426202 and 11426200), the Zhejiang Provincial Natural Science Foundation of China (Nos. LQ12A01020, LQ13F030010, LY15F030013, and LQ14G010004), the Ministry of Education, Humanities and Social Sciences Research Project of China (No. 13YJC910011) and the Science Foundation of Chongqing Municipal Commission of Science and Technology (Grant No. CSTC201 4jcyjA40011) and Scientific and Technological Research Program of Chongqing Municipal Education Commission (Grant No. KJ1400513).

References

1. Cortes C, Vapnik V (1995) Support vector networks. Mach Learn 20(3):273–297

2. Burges C (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Discov 2:121–167
3. Noble W (2004) Support vector machine applications in computational biology. In: Schöelkopf B, Tsuda K, Vert J-P (eds) Kernel methods in computational biology. MIT Press, Cambridge, pp 71–92
4. Li Y, Shao Y, Jing L, Deng N (2011) An efficient support vector machine approach for identifying protein s-nitrosylation sites. Protein Pept Lett 18(6):573–587
5. Li Y, Shao Y, Deng N (2011) Improved prediction of palmitoylation sites using PWMs and SVM. Protein Pept Lett 18(2):186–193(8)
6. Vapnik V (1998) Statistical learning theory. Wiley, New York
7. Suykens J, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
8. Fung G, Mangasarian O (2005) Multicategory proximal support vector machine classifiers. Mach Learn 59:77–97
9. Mangasarian O, Wild E (2006) Multisurface proximal support vector classification via generalize eigenvalues. IEEE Trans Pattern Anal Mach Intell 28(1):69–74
10. Jayadeva R, Khemchandani R, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910
11. Qi Z, Tian Y, Shi Y (2015) Successive overrelaxation for laplacian support vector machine. IEEE Trans Neural Netw Learn Syst 26(4):674–683
12. Tanveer M (2015) Robust and sparse linear programming twin support vector machines. Cogn Comput 7(1):137–149
13. Shao Y, Deng N, Chen W, Wang Z (2013) Improved generalized eigenvalue proximal support vector machine. IEEE Signal Process Lett 20(3):213–216
14. Kumar M, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36(4):7535–7543
15. Shao Y, Zhang C, Wang X, Deng N (2011) Improvements on twin support vector machines. IEEE Trans Neural Netw 22(6):962–968
16. Shao Y, Deng N (2012) A coordinate descent margin based-twin support vector machine for classification. Neural Netw 25:114–121
17. Qi Z, Tian Y, Shi Y (2012) Laplacian twin support vector machine for semi-supervised classification. Neural Netw 35:46–53
18. Shao Y, Chen W, Deng NY (2014) Nonparallel hyperplane support vector machine for binary classification problems. Inf Sci 263(1):22–35
19. Tian Y, Qi Z, Ju X (2014) Nonparallel support vector machine for pattern classification. IEEE Trans Cybern 44(7):1067–1079
20. Wang Z, Shao Y, Bai L, Deng N (2014) Twin support vector machine for clustering. IEEE Trans Neural Netw Learn Syst. doi:10.1109/TNNLS.2014.2379930
21. Ye Q, Zhao C, Ye N, Chen Y (2010) Multi-weight vector projection support vector machines. Pattern Recognit Lett 31(13):2006–2011
22. Chen X, Yang J, Ye Q, Liang J (2011) Recursive projection twin support vector machine via within-class variance minimization. Pattern Recognit 44(10):2643–2655
23. Shao Y, Wang Z, Chen W, Deng N (2013) A regularization for the projection twin support vector machine. Knowl-Based Syst 37:203–210
24. Shao Y, Deng N, Yang Z (2012) Least squares recursive projection twin support vector machine for classification. Pattern Recognit 45(6):2299–2307
25. Weston J, Watkins C (1998) Multi-class support vector machines. Technical report CSD-TR-98-04
26. Schwenker F (2000) Hierarchical support vector machines for multi-class pattern recognition. In: Fourth international conference on knowledge-based intelligent information engineering systems & allied technologies, vol 2, pp 561–565
27. Lee Y, Lin Y, Wahba G (2004) Multicategory support vector machines: theory and application to the classification of microarray data and satellite radiance data. J Am Stat Assoc 99(465):67–81
28. Yang Z, Shao Y, Zhang X (2013) Multiple birth support vector machine for multi-class classification. Neural Comput Appl 22(1Suppl):153–161
29. Li C, Huang Y, Wu H, Shao Y, Yang Z (2014) Multiple recursive projection twin support vector machine for multi-class classification. Int J Mach Learn Cybern. doi:10.1007/s13042-014-0289-2
30. Golub G, Van Loan C (1996) Matrix computations, 3rd edn. Johns Hopkins University Press, Baltimore
31. Belhumeur P, Hespanha J, Kriegman D (1997) Eigenfaces vs. fisherfaces: recognition using class specific linear projection. IEEE Trans Pattern Anal Mach Intell 19(7):711–720
32. Yang J (2003) Why can LDA be performed in PCA transformed space? Pattern Recognit 36(2):563–566
33. Bishop C (2006) Pattern recognition and machine learning. Springer, New York
34. Tao Q, Chu D, Wang J (2008) Recursive support vector machines for dimensionality reduction. IEEE Trans Neural Netw 19(1):189–193
35. Lee YJ, Mangasarian O (2001) RSVM: reduced support vector machines. Technical report 00-07. Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison (2001)
36. Blake C, Merz C (1998) UCI repository for machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html
37. Chang C, Lin C (2011) LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets
38. Chang C, Lin C (2011) LIBSVM : a library for support vector machines. ACM Trans Intell Syst Technol 2(27):1–27
39. Duda R, Hart P, Stork D (2012) Pattern classification, 2nd edn. Wiley, New York
40. Jin Z, Yang J, Hu Z, Lou Z (2001) Face recognition based on the uncorrelated discriminant transformation. Pattern Recognit 34:1405–1416