Neural Comput & Applic (2014) 24:1207–1220
DOI 10.1007/s00521-012-1306-6

ORIGINAL ARTICLE

Twin support vector hypersphere (TSVH) classifier for pattern recognition

Xinjun Peng · Dong Xu

Received: 14 January 2012 / Accepted: 7 December 2012 / Published online: 5 February 2013
© Springer-Verlag London 2013

Abstract Motivated by the support vector data description, a classical one-class support vector machine, and the twin support vector machine classifier, this paper formulates a twin support vector hypersphere (TSVH) classifier, a novel binary support vector machine (SVM) classifier that determines a pair of hyperspheres by solving two related SVM-type quadratic programming problems, each of which is smaller than that of a conventional SVM, which means that this TSVH is more efficient than the classical SVM. In addition, the TSVH successfully avoids matrix inversion compared with the twin support vector machine, which indicates learning algorithms of the SVM can be easily extended to this TSVH. Computational results on several synthetic as well as benchmark data sets indicate that the proposed TSVH is not only faster, but also obtains better generalization.

Keywords Machine learning · Pattern recognition · Support vector machine · Nonparallel hyperplanes · Hypersphere

1 Introduction

The support vector machine (SVM) is an excellent tool for binary data classification [1–4]. This learning strategy, introduced by Vapnik [3, 4], is a principled and very powerful method in machine learning algorithms. Within a few years after its introduction, the SVM has already outperformed most other systems in a wide variety of applications. These include a wide spectrum of research areas, ranging from pattern recognition [5, 6], text categorization [7], biomedicine [8], brain–computer interface [9], and financial applications [10].

The theory of SVM proposed by Vapnik et al. is based on the structural risk minimization (SRM) principle [1–4]. In its simplest form, the SVM for linearly separable two-class problems finds an optimal hyperplane that maximizes the margin between the two classes. The hyperplane is obtained by solving a dual quadratic programming problem (QPP). For nonlinearly separable cases, the input vectors are first mapped into a high dimensional feature space by using a nonlinear kernel function [1, 4]. A linear classifier is then implemented in that feature space to classify the data.

One of the main challenges for the classical SVM is the large computational complexity of the QPP. In addition, the performance of a trained SVM also depends on the optimal parameter set, which is usually found by cross-validation on a training set. The long training time of QPP not only causes the SVM to take a long time to train on a large database, but also prevents the SVM from locating the optimal parameter set from a very fine grid of parameters over a large span. To reduce the learning complexity of SVM, various algorithms and versions of SVM have been reported by many researchers with comparable classifications ability, including the Chunking algorithm [11], decomposition method [12], sequential minimal optimization (SMO) approach [13, 14], least squares support vector machine (LS-SVM) [15, 16], and geometric algorithms [18–20].

All the above classifiers discriminate a point by determining in which half-space it lies. Mangasarian and Wild [21] proposed a nonparallel-plane classifier for binary data

X. Peng (✉) · D. Xu
Department of Mathematics, Shanghai Normal University, Shanghai 200234, People's Republic of China
e-mail: xjpeng999@gmail.com

X. Peng
Scientific Computing Key Laboratory of Shanghai Universities, Shanghai 200234, People's Republic of China

classification, named the generalized eigenvalue proximal support vector machine (GEPSVM). In this approach, data points of each class are proximal to one of two nonparallel planes. The nonparallel planes are eigenvectors corresponding to the smallest eigenvalues of two related generalized eigenvalue problems. The GEPSVM is very fast as it solves two generalized eigenvalue problems of the order of input space dimension. But the performance of it is only comparable with the classical SVM and in many cases, it gives low classification accuracies. Recently, Jayadeva et al. [22] proposed a twin support vector machine (TSVM) classifier for binary data classification. The TSVM aims at generating two nonparallel hyperplanes such that each one is closer to one class and is at least one far from the other class for any given binary data set. The strategy of solving a pair of smaller sized QPPs instead of a large one as in classical SVMs makes the learning speed of TSVM be approximately four times faster than classical SVM. In terms of generalization, the TSVM favorably compares with classical SVM. Some extensions to the TSVM include the least squares TSVM (LS-TSVM) [23], smooth TSVM [24], nonparallel-plane proximal classifier (NPPC) [25, 26], geometric algorithms [27], localized TSVM (LCTSVM) [28], twin support vector regression (TSVR) [29, 30], twin parametric-margin SVM (TPMSVM) [31], and twin parametric insensitive support vector regressions (TPISVR) [32].

The experimental results in [22] show that the TSVM obtains four times faster learning speed than a classical SVM. This is because the TSVM differs from the SVM in one fundamental way, in which a pair of smaller sized QPPs is solved, whereas in the classical SVM, only a single QPP is solved. In addition, the TSVM compares favorably with the SVM and GEPSVM in terms of generalization performance. However, there exists some shortcomings in the TSVM. First, for the nonlinear TSVM, the objective functions of its dual QPPs require inversion of matrix of size $(l + 1) \times (l + 1)$ twice, where l is the number of training samples, which increases the computational cost for optimizing them. On the other hand, in order to extend some efficient SVM algorithms on the TSVM, such as the SMO and decomposition algorithm, it needs to cache the inverse matrices, which leads the TSVM to be not suitable for large-scale problems. Second, the TSVM uses a pair of nonparallel hyperplanes to depict the characterizations of two classes of samples, such that each hyperplane is closest to data points of the corresponding class and is at a distance of at least 1 from the data points of the other one. It is obviously that this description is not reasonable for general cases; for instance, it is impossible to use two planes to reflect the binary classification samples coming from two different Gaussian distributions. Third, unlike the classical SVM, the TSVM only aims at minimizing the empirical risks of training samples such that in each QPP, the L_2-norm empirical risks of samples of one class and the L_1-norm empirical risks of samples of the other class are minimized simultaneously with a trade-off parameter. A direct consequence is to appear the over-fitting phenomenon on the nonparallel hyperplanes, which reduce the generalization performance.

In this paper, we propose a new hypersphere classifier, termed the twin support vector hypersphere (TSVH), for binary pattern recognition. Our TSVH is motivated by the support vector data description (SVDD) [33, 34], a classical one-class support vector machine (OCSVM) for unsupervised learning (the readers can find the introduction of OCSVM at [35] etc). Basically, the TSVH aims at generating two hyperspheres such that each hypersphere contains as many as possible samples of one class and is as far as possible from those of the other class. Similar to the TSVM, the TSVH solves two smaller sized QPPs instead of a large one as we do in the traditional SVM. There exists some merits in the TSVH compared with the TSVM. First, the formulation of TSVH is totally different from that of the TSVM. Specifically, the inversions of matrix of size $(l + 1) \times (l + 1)$ twice in the dual QPPs of TSVM are avoided, which cause the TSVH to obtain a lower learning cost and can be easily extended to large-scale problems with efficient learning algorithms. Second, the TSVH uses a pair of hyperspheres to describe the training samples, such as each hypersphere covers as many as possible samples of the one class, whereas is far from the samples of the other class. This strategy is more reasonable than the TSVM for most real cases, which means TSVH can deal with more general pattern recognition problems than TSVM. Computational comparisons with the TSVM and SVM in terms of classification accuracy and learning time are made on several artificial and UCI data sets, showing that the proposed TSVH is not only faster than the other two methods, but also obtains comparable generalization.

The remainder of this paper is organized as follows: Sect. 2 briefly reviews the SVM, SVDD, and TSVM. Section 3 introduces the novel twin support vector hypersphere. Section 4 discusses the learning algorithm for the proposed TSVH. Experiments on several artificial and benchmark data sets are discussed in Sect. 5. Finally, Sect. 6 summarizes and concludes the paper.

2 Background

In this section, we briefly introduce the classical SVM, SVDD, and TSVM. Without the loss of generalization, we consider a classification problem with the data set $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}$, where $\boldsymbol{x}_i \in \mathcal{X} \subset \mathcal{R}^d$ and $y_i \in$

$\{-1, 1\}$. Further, we denote by $\mathcal{I}^{\pm}$ the sets of indices i such that $y_i = \pm 1$ and $\mathcal{I} = \mathcal{I}^+ \bigcup \mathcal{I}^-$, and denote by $D^{\pm}$ the positive and negative set, i.e., $D^{\pm} = \{\boldsymbol{x}_i | i \in \mathcal{I}^{\pm}\}$.

2.1 Support vector machine

Generally, the SVM finds the best (maximal margin) separating hyperplane $H(\boldsymbol{w}, b)$ between two classes of samples in the feature space $\mathcal{H}$

$$H(\boldsymbol{w}, b) : \boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}) + b = 0 \tag{1}$$

maximizing the total (interclass) margin $2/||\boldsymbol{w}||$ and satisfying $y_i(\boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}_i) + b) - 1 \geq 0$, $i \in \mathcal{I}$, where $\boldsymbol{\varphi}(\cdot) : \mathcal{X} \to \mathcal{H}$ maps $\mathcal{X}$ into $\mathcal{H}$, $\boldsymbol{w} \in \mathcal{H}$, and $||\cdot||$ is the L_2-norm. For the linear case, we have $\boldsymbol{\varphi}(\boldsymbol{x}) = \boldsymbol{x}$. Under the Mercer theorem [3, 4, 36], it is possible to use a kernel $k(\boldsymbol{u}, \boldsymbol{v})$ to represent the inner product in $\mathcal{H}$, i.e., $k(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{\varphi}(\boldsymbol{u})^T \boldsymbol{\varphi}(\boldsymbol{v})$, such as the Gaussian kernel $k(\boldsymbol{u}, \boldsymbol{v}) = \exp\{-\gamma ||\boldsymbol{u} - \boldsymbol{v}||^2\}$ with $\gamma > 0$. To fit practical applications, one way is to add a slack variable ξ_i for each $\boldsymbol{x}_i$, which allows a controlled violation of the constraint. Therefore, SVM can be expressed as the following mathematical model:

$$\begin{aligned} &\min \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i\in\mathcal{I}} \xi_i \\ &\text{s.t. } y_i\left(\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}_i) + b\right) \geq 1 - \xi_i, \\ &\quad \xi_i \geq 0,\ i \in \mathcal{I}, \end{aligned} \tag{2}$$

which is a QPP with linear inequalities in $\mathcal{H}$, where $C > 0$ is the pre-specified regularization factor given by users. By introducing the Lagrangian coefficients α_i's, we derive its dual QPP as following:

$$\begin{aligned} &\max \quad \sum_{i\in\mathcal{I}} \alpha_i - \frac{1}{2}\sum_{i,j\in I} \alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\text{s.t.} \quad 0 \leq \alpha_i \leq C,\ i \in \mathcal{I}, \\ &\sum_{i\in\mathcal{I}} \alpha_i y_i = 0. \end{aligned} \tag{3}$$

After optimizing this dual QPP, we obtain the following decision function:

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i\in\mathcal{I}} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b\right), \tag{4}$$

where $\boldsymbol{w}$ in $H(\boldsymbol{w}, b)$ is derived by the Karush–Kuhn–Tucker (KKT) optimality conditions, which is:

$$\boldsymbol{w} = \sum_{i\in\mathcal{I}} y_i\alpha_i\boldsymbol{\varphi}(\boldsymbol{x}_i) = \sum_{i\in\mathcal{I}^+} \alpha_i\boldsymbol{\varphi}(\boldsymbol{x}_i) - \sum_{j\in\mathcal{I}^-} \alpha_j\boldsymbol{\varphi}(\boldsymbol{x}_j). \tag{5}$$

Note that optimizing the dual QPP (3) is time-consuming for large-scale problems because of enormous matrix storages and intensive matrix operations. Thus, a major research topic related to SVM is to focus on the fast learning aspect, e.g., [11, 13, 14, 17–20].

2.2 Twin support vector machine

The TSVM is a binary classifier that does classification using two nonparallel hyperplanes instead of a single hyperplane in the classical SVM [22], which are obtained by solving two smaller sized QPPs.

For the linear case, the TSVM finds the following pair of nonparallel positive and negative hyperplanes in $\mathcal{R}^d$:

$$\boldsymbol{w}_+^T\boldsymbol{x} + b_+ = 0, \quad \boldsymbol{w}_-^T\boldsymbol{x} + b_- = 0, \tag{6}$$

such that each hyperplane is closer to the samples of one class and is as far as possible from those of the other class. A new point is assigned to class $+1$ (positive) or -1 (negative) depending upon its proximity to the above two nonparallel hyperplanes. Formally, the linear TSVM solves the following two QPPs for finding the positive and negative hyperplanes, respectively:

$$\begin{aligned} &\min \frac{1}{2}\sum_{i\in\mathcal{I}^+} \left(\boldsymbol{w}_+^T\boldsymbol{x}_i + b_+\right)^2 + C_1\sum_{j\in\mathcal{I}^-} \xi_j \\ &\text{s.t. } -\boldsymbol{w}_+^T\boldsymbol{x}_j - b_+ \geq 1 - \xi_j, \\ &\quad \xi_j \geq 0,\ j \in \mathcal{I}^-, \end{aligned} \tag{7}$$

$$\begin{aligned} &\min \frac{1}{2}\sum_{j\in\mathcal{I}^-} \left(\boldsymbol{w}_-^T\boldsymbol{x}_j + b_-\right)^2 + C_2\sum_{i\in\mathcal{I}^+} \xi_i \\ &\text{s.t. } \boldsymbol{w}_-^T\boldsymbol{x}_i + b_- \geq 1 - \xi_i, \\ &\quad \xi_i \geq 0,\ i \in \mathcal{I}^+, \end{aligned} \tag{8}$$

where C_1, $C_2 > 0$ are the pre-specified trade-off factors, and $\xi_j, j \in \mathcal{I}^-$, $\xi_i, i \in \mathcal{I}^+$ are the Slack variables. By introducing the Lagrangian multipliers, we obtain the following dual QPPs[1]:

$$\begin{aligned} &\max \quad \sum_{j\in\mathcal{I}^-} \alpha_j - \frac{1}{2}\sum_{j_1,j_2\in\mathcal{I}^-} \alpha_{j_1}\alpha_{j_2}\boldsymbol{z}_{j_1}^T\left(\sum_{i\in\mathcal{I}^+} \boldsymbol{z}_i\boldsymbol{z}_i^T\right)^{-1}\boldsymbol{z}_{j_2} \\ &\text{s.t.} \quad 0 \leq \alpha_j \leq C_1,\ j \in \mathcal{I}^-, \end{aligned} \tag{9}$$

[1] If $\sum_{i\in\mathcal{I}^+} \boldsymbol{z}_i\boldsymbol{z}_i^T$ and $\sum_{j\in\mathcal{I}^-} \boldsymbol{z}_j\boldsymbol{z}_j^T$ are ill-conditioning, we can add the regularization terms $\epsilon_i E$, $\epsilon_i > 0$, here, E is an identity matrix of appropriate dimension. Remark that these two added regularization terms are equivalent to adding two regularization terms $\frac{\epsilon_i}{2}||(\boldsymbol{w}_{\pm}||^2 + b_{\pm}^2), i = 1, 2$, into the objective functions of TSVM, which make TWSVMs be more theoretical sound [37, 38]. In the view point of regression, they make the nonparallel hyperplanes be more smooth and robust.

are the centers and

$$\max \sum_{i\in\mathcal{I}^+}\alpha_i-\frac{1}{2}\sum_{i_1,i_2\in\mathcal{I}^+}\alpha_{i_1}\alpha_{i_2}z_{i_1}^T\left(\sum_{j\in\mathcal{I}^-}z_jz_j^T\right)^{-1}z_{i_2} \quad (10)$$
$$\text{s.t. } 0\le\alpha_i\le C_2,\ i\in\mathcal{I}^+,$$

where $z_k=(\boldsymbol{x}_k;1)$, $k\in\mathcal{I}$. After optimizing (9) and (10), we obtain the augmented vectors of the two non-parallel hyperplanes, which are:

$$\boldsymbol{v}_+^T=(\boldsymbol{w}_+^T,b_+)=-\sum_{j\in\mathcal{I}^-}\alpha_jz_j^T\left(\sum_{i\in\mathcal{I}^+}z_iz_i^T\right)^{-1}, \quad (11)$$

$$\boldsymbol{v}_-^T=(\boldsymbol{w}_-^T,b_-)=\sum_{i\in\mathcal{I}^+}\alpha_iz_i^T\left(\sum_{j\in\mathcal{I}^-}z_jz_j^T\right)^{-1}. \quad (12)$$

A new data point $\boldsymbol{x}$ is then assigned to the class + or −, depending on which of the two hyperplanes given by (6) it lies closest to. Thus,

$$f(\boldsymbol{x})=\arg\min_{+,-}\{d_\pm(\boldsymbol{x})\},$$

where

$$d_\pm(\boldsymbol{x})=\frac{|\boldsymbol{w}_\pm^T\boldsymbol{x}+b_\pm|}{||\boldsymbol{w}_\pm||}.$$

For the nonlinear case, the nonparallel positive and negative hyperplanes are expressed as following:

$$\sum_{k\in\mathcal{I}}w_{k,+}k(\boldsymbol{x}_k,\boldsymbol{x})+b_+=0 \quad \text{and} \quad \sum_{k\in\mathcal{I}}w_{k,-}k(\boldsymbol{x}_k,\boldsymbol{x})+b_-=0, \quad (13)$$

with $\boldsymbol{w}_\pm=(w_{1,\pm};w_{2,\pm};\ldots;w_{l,\pm})$ and $k(\boldsymbol{u},\boldsymbol{v})$ is some kernel. It is easy to obtain the primal QPPs for this case, which are similar to (11) and (12). If we define $z_k=(k(\boldsymbol{x}_1,\boldsymbol{x}_k);\ldots;k(\boldsymbol{x}_l,\boldsymbol{x}_k);1)$ for each $k\in\mathcal{I}$, then we have the same dual QPPs as (9) and (10) and the same augmented vectors as (11) and (12) for (13).

The QPPs (9) and (10) have lower complexity than (3) since they have only $l^\mp=|\mathcal{I}^\mp|$ parameters, while (3) has $l=l^++l^-$ parameters, which causes the TSVM to be approximately four times faster than the classical SVM. This is because the complexity of SVM is no more than $\mathcal{O}(l^3)$, while the TSVM solves two QPPs, each of which is roughly of size ($l/2$). Thus, the ratio of runtime is approximately 4. It is necessary to point out that, in order to optimize (9) and (10), the linear TSVM requires inversion of matrix of size $(d+1)\times(d+1)$ twice, whereas the nonlinear TSVM requires inversion of matrix of size $(l+1)\times(l+1)$ twice. Further, the samples with $0<\alpha_j\le C_1, j\in\mathcal{I}^-$ or $0<\alpha_i\le C_2, i\in\mathcal{I}^+$ are defined as SVs, since they are significant in determining the two hyperplanes (6). The experiments show that the TSVM obtains good generalization performance and faster learning speed than the classical SVM. However, as the above discussion, there still exists some shortcomings in the TSVM, which reduce the application ranges and generalization of TSVM.

2.3 Support vector data description

An one-class problem is usually understood as computing a binary function that is supposed to capture regions in input space where the probability density lies, that is, a function such that most of the samples will lie in the region where the function is nonzero (see Refs. [33–35]). In the viewpoint of learning, an one-class problem is an unsupervised one, where the samples are unlabeled or the label of each sample is thought to be itself. Over the last decade, the SVM has already been generalized to solve one-class problems. One of the popular OCSVM is the SVDD proposed by Tax and Duin [33, 34].

The SVDD identifies a sphere with minimum volume that captures the given normal dataset. The sphere volume is characterized with its center $\boldsymbol{c}$ and radius R in some feature space. Minimization of the volume is achieved by minimizing R^2, which represents the structural error:

$$\min \quad R^2 \quad \text{s.t.} \quad ||\boldsymbol{\varphi}(\boldsymbol{x}_i)-\boldsymbol{c}||^2\le R^2,\ \forall i. \quad (14)$$

The above constraints do not allow any point to fall outside of the hypersphere. In order to make provision within the model for potential outliers within the training set, a penalty cost function is introduced as follows (for samples that lie outside of the hypersphere):

$$\min \quad R^2+C\sum_i\xi_i \quad \text{s.t.} \quad ||\boldsymbol{\varphi}(\boldsymbol{x}_i)-\boldsymbol{c}||^2\le R^2+\xi_i,\ \forall i. \quad (15)$$

where C is the coefficient of penalty for each outlier and ξ_i is the distance between the ith sample and the hypersphere. This is a quadratic optimization problem and can be solved efficiently by introducing Lagrange multipliers for constraints. Formally, it can be solved by optimizing the following dual problem:

$$\max \quad \sum_i\alpha_ik(\boldsymbol{x}_i,\boldsymbol{x}_i)-\sum_{i,j}\alpha_i\alpha_jk(\boldsymbol{x}_i,\boldsymbol{x}_j) \quad \text{s.t.} \quad 0\le\alpha_i\le C,\ \forall i, \quad \sum_i\alpha_i=1. \quad (16)$$

This problem can be solved rather easily using well-established quadratic programming algorithms and will lead to the representation of "normal" data. The assessment of whether a data point is inside or outside

the SVDD hypersphere is based on the sign of the following function:

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(R^2 - ||\boldsymbol{\varphi}(\boldsymbol{x}) - \boldsymbol{c}||^2\right) = \operatorname{sgn}\left(R^2 - \sum_{i,j}\alpha_i\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j) + 2\sum_i \alpha_i k(\boldsymbol{x}_i,\boldsymbol{x}) - k(\boldsymbol{x},\boldsymbol{x})\right), \tag{17}$$

where R^2 is computed according to the KKT conditions, and the center $\boldsymbol{c}$ is

$$\boldsymbol{c} = \sum_i \alpha_i \boldsymbol{\varphi}(\boldsymbol{x}_i). \tag{18}$$

Positive (negative) sign implies that the distance of the data point to the center of the hyper-sphere is less (greater) than the radius of the hypersphere.

3 Twin support vector hypersphere

In this section, we introduce a novel approach to SVM classification which we have termed the twin support vector hypersphere classifier.

As mentioned earlier, the TSVH is similar to the TSVM in spirit, as it also obtains two QPPs for binary pattern recognition, and each QPP has the SVM-type formulation, except that not all patterns appear in the constraints of either problem at the same time. However, it is based on an entirely different fact that the samples of different classes get clustered, which can be covered by two hyperspheres, respectively, that is, the samples of one class are covered by one hypersphere, whereas the samples of the other class are as possible as far from this hypersphere. In short, it is the combination of the idea of TSVM and SVDD.

The TSVH classifier is obtained by solving the following pair of QPPs:

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i\in\mathcal{I}^+}||\boldsymbol{\varphi}(\boldsymbol{x}_i) - \boldsymbol{c}_+||^2 - \nu_1 R_+^2 + C_1\sum_{j\in\mathcal{I}^-}\xi_j \\ \text{s.t.} \quad & \left|\left|\boldsymbol{\varphi}(\boldsymbol{x}_j) - \boldsymbol{c}_+\right|\right|^2 \geq R_+^2 - \xi_j, \\ & R_+^2 \geq 0,\ \xi_j \geq 0,\ j \in \mathcal{I}^-, \end{aligned} \tag{19}$$

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{j\in\mathcal{I}^-}\left|\left|\boldsymbol{\varphi}(\boldsymbol{x}_j) - \boldsymbol{c}_-\right|\right|^2 - \nu_2 R_-^2 + C_2\sum_{i\in\mathcal{I}^-}\xi_i \\ \text{s.t.} \quad & ||\boldsymbol{\varphi}(\boldsymbol{x}_i) - \boldsymbol{c}_-||^2 \geq R_-^2 - \xi_i, \\ & R_-^2 \geq 0,\ \xi_i \geq 0,\ i \in \mathcal{I}^+, \end{aligned} \tag{20}$$

where $C_1, C_2 > 0$ and $\nu_1, \nu_2 > 0$ are the pre-specified penalty factors, and $\boldsymbol{c}_\pm \in \mathcal{H}$ and $R_\pm$ are the centers and radiuses of corresponding hyperspheres, respectively.

This model finds two hyperspheres, one for each class, and classifies points according to which hypersphere a given point is closest to. The first term in the objective function of (19) or (20) is the sum of squared distances from the center of hypersphere to points of one class. Therefore, minimizing it tends to keep the center of hypersphere close to the points of one class. The second term maximizes the squared radius of this hypersphere, which makes this hypersphere be as large as possible. The first constraints require the center of hypersphere to be at a distance of at least its radius from points of the opposite class; a set of error variables is used to measure the errors wherever the points of opposite class are covered by this hypersphere, that is, the distances from the points to the center are less than the radius. The last term of the objective function of (19) or (20) minimizes the sum of error variables, thus attempting to minimize misclassification due to points belonging to the opposite class.

In short, similar to the TSVM, the TSVH is comprised of a pair of QPPs such that, in each QPP, the objective function corresponds to a particular class and the constraints are determined by the samples of the other class. Thus, the TSVH gives rise to two smaller sized QPPs compared with a single large QPP in the classical SVM, leading the TSVH to be approximately four times faster than the classical SVM for learning its hyperspheres. However, unlike the formulation of TSVM that around each hyperplane, the data points of the corresponding class get clustered, Our TSVH, in the spirit of the SVDD, uses a pair of hyperspheres to depict the two classes. In the problem (19), the positive samples are covered by a hypersphere and clustered around the center $\boldsymbol{c}_+$, and in the problem (20), the negative samples are covered by a hypersphere and clustered around the center $\boldsymbol{c}_-$. For most cases, this strategy is more reasonable than that of the TSVM.

The corresponding Lagrangian function of (19) is

$$\begin{aligned} L(\boldsymbol{c}_+, R_+^2, \xi, \alpha, \boldsymbol{r}, s) = {} & \frac{1}{2}\sum_{i\in\mathcal{I}^+}||\boldsymbol{\varphi}(\boldsymbol{x}_i) - \boldsymbol{c}_+||^2 - \nu_1 R_+^2 + C_1\sum_{j\in\mathcal{I}^-}\xi_j \\ & - \sum_{j\in\mathcal{I}^-}\alpha_j\left(\left|\left|\boldsymbol{\varphi}(\boldsymbol{x}_j) - \boldsymbol{c}_+\right|\right|^2 - R_+^2 + \xi_j\right) \\ & - \sum_{j\in\mathcal{I}^-} r_j\xi_j - sR_+^2, \end{aligned} \tag{21}$$

where α_j, r_j, $j \in \mathcal{I}^-$ and s are the Lagrangian multipliers, $\boldsymbol{\alpha} = (\alpha_{j_1}; \alpha_{j_2}; \ldots; \alpha_{j_{l^-}})$, $\boldsymbol{r} = (r_{j_1}; r_{j_2}; \ldots; r_{j_{l^-}})$, and $\xi = (\xi_{j_1}; \xi_{j_1}; \ldots; \xi_{j_{l^-}})$, $j_k \in \mathcal{I}^-$, $k = 1, \ldots, l^-$, respectively. Differentiating the Lagrangian function (21) with respect to $\boldsymbol{c}_+$, R_+^2, and ξ_j, $j \in \mathcal{I}^-$ yields the following KKT necessary and sufficient optimality conditions:

$$\frac{\partial L}{\partial \boldsymbol{c}_+} = -\sum_{i\in\mathcal{I}^+}(\boldsymbol{\varphi}(\boldsymbol{x}_i)-\boldsymbol{c}_+)+2\sum_{j\in\mathcal{I}^-}\alpha_j\big(\boldsymbol{\varphi}(\boldsymbol{x}_j)-\boldsymbol{c}_+\big)=0$$
$$\rightarrow \boldsymbol{c}_+ = \frac{1}{l^+ - 2\sum\limits_{j\in\mathcal{I}^-}\alpha_j}\left(\sum_{i\in\mathcal{I}^+}\boldsymbol{\varphi}(\boldsymbol{x}_i)-2\sum_{j\in\mathcal{I}^-}\alpha_j\boldsymbol{\varphi}(\boldsymbol{x}_j)\right), \tag{22}$$

$$\frac{\partial L}{\partial R_+^2} = -\nu_1+\sum_{j\in\mathcal{I}^-}\alpha_j - s = 0 \rightarrow \sum_{j\in\mathcal{I}^-}\alpha_j \geq \nu_1, \tag{23}$$

$$\frac{\partial L}{\partial \xi_j} = C_1-\alpha_j-r_j = 0 \rightarrow 0\leq\alpha_j\leq C_1,\quad j\in\mathcal{I}^-, \tag{24}$$

$$\big|\big|\boldsymbol{\varphi}(\boldsymbol{x}_j)-\boldsymbol{c}_+\big|\big|^2 \geq R_+^2-\xi_j,\quad j\in\mathcal{I}^-, \tag{25}$$

$$\alpha_j\Big(\big|\big|\boldsymbol{\varphi}(\boldsymbol{x}_j)-\boldsymbol{c}_+\big|\big|^2-R_+^2+\xi_j\Big)=0,\quad \alpha_j\geq 0,\quad j\in\mathcal{I}^-, \tag{26}$$

$$r_j\xi_j = 0,\quad \xi_j\geq 0,\quad r_j\geq 0,\quad j\in\mathcal{I}^-, \tag{27}$$

$$sR_+^2 = 0,\quad R_+^2\geq 0,\quad s\geq 0. \tag{28}$$

Since $R_+^2 > 0$ holds in the optimality result of problem (19) if given a suitable parameter ν_1, then from (23) and (28), we have

$$\sum_{j\in\mathcal{I}^-}\alpha_j = \nu_1. \tag{29}$$

Substituting (29) into (22) leads to

$$\boldsymbol{c}_+ = \frac{1}{l^+-2\nu_1}\left(\sum_{i\in\mathcal{I}^+}\boldsymbol{\varphi}(\boldsymbol{x}_i)-2\sum_{j\in\mathcal{I}^-}\alpha_j\boldsymbol{\varphi}(\boldsymbol{x}_j)\right), \tag{30}$$

which obtains a by-product for determining ν_1 that $2\nu_1 < l^+$. Substituting (23), (24), and (30) into (21), we obtain the dual QPP of (19), which is:

$$\begin{aligned}\max\quad & \frac{1}{2}\sum_{i\in\mathcal{I}^+}k(\boldsymbol{x}_i,\boldsymbol{x}_i)-\frac{1}{2(l^+-2\nu_1)}\sum_{i_1,i_2\in\mathcal{I}^+}k(\boldsymbol{x}_{i_1},\boldsymbol{x}_{i_2})\\ & -\sum_{j\in\mathcal{I}^-}\alpha_j k(\boldsymbol{x}_j,\boldsymbol{x}_j)\\ & -\frac{2}{l^+-2\nu_1}\sum_{j_1,j_2\in\mathcal{I}^-}\alpha_{j_1}\alpha_{j_2}k(\boldsymbol{x}_{j_1},\boldsymbol{x}_{j_2})+\frac{2}{l^+-2\nu_1}\\ & \sum_{i\in\mathcal{I}^+,j\in\mathcal{I}^-}\alpha_j k(\boldsymbol{x}_j,\boldsymbol{x}_i)\\ \text{s.t.}\quad & \sum_{j\in\mathcal{I}^-}\alpha_j=\nu_1,\ 0\leq\alpha_j\leq C_1,\ j\in\mathcal{I}^-.\end{aligned} \tag{31}$$

Defining $t_1 = \frac{2}{l^+-2\nu_1}$ and discarding the constant items in the objective function of (31), the dual QPP (31) can be simplified as following:

$$\begin{aligned}\max\quad & -t_1\sum_{j_1,j_2\in\mathcal{I}^-}\alpha_{j_1}\alpha_{j_2}k(\boldsymbol{x}_{j_1},\boldsymbol{x}_{j_2})\\ & +\sum_{j\in\mathcal{I}^-}\alpha_j\left(t_1\sum_{i\in\mathcal{I}^+}k(\boldsymbol{x}_j,\boldsymbol{x}_i)-k(\boldsymbol{x}_j,\boldsymbol{x}_j)\right)\\ \text{s.t.}\quad & \sum_{j\in\mathcal{I}^-}\alpha_j=\nu_1,\ 0\leq\alpha_j\leq C_1,\ j\in\mathcal{I}^-.\end{aligned} \tag{32}$$

Similarly, we consider (20) and obtain its dual QPP as

$$\begin{aligned}\max\quad & -t_2\sum_{i_1,i_2\in\mathcal{I}^+}\beta_{i_1}\beta_{i_2}k(\boldsymbol{x}_{i_1},\boldsymbol{x}_{i_2})\\ & +\sum_{i\in\mathcal{I}^+}\beta_i\left(t_2\sum_{j\in\mathcal{I}^-}k(\boldsymbol{x}_i,\boldsymbol{x}_j)-k(\boldsymbol{x}_i,\boldsymbol{x}_i)\right)\\ \text{s.t.}\quad & \sum_{i\in\mathcal{I}^+}\beta_i=\nu_2,\ 0\leq\beta_i\leq C_2,\ i\in\mathcal{I}^+.\end{aligned} \tag{33}$$

Here, $t_2 = \frac{2}{l^--2\nu_2}$, β_i, $i\in\mathcal{I}^+$ are the Lagrangian multipliers, and the center $\boldsymbol{c}_-$ is

$$\boldsymbol{c}_- = \frac{1}{l^--2\nu_2}\left(\sum_{j\in\mathcal{I}^-}\boldsymbol{\varphi}(\boldsymbol{x}_j)-2\sum_{i\in\mathcal{I}^+}\beta_i\boldsymbol{\varphi}(\boldsymbol{x}_i)\right). \tag{34}$$

Next, define the index sets

$$\begin{aligned}I_R^+ &= \{i|0<\beta_i<C_2,\ i\in\mathcal{I}^+\}\\ \text{and}\quad I_R^- &= \{j|0<\alpha_j<C_1,\ j\in\mathcal{I}^-\},\end{aligned} \tag{35}$$

according to the KKT conditions, we derive the square radiuses $R^2{}_\pm$ as

$$R_+^2 = \frac{1}{|I_R^-|}\sum_{j\in I_R^-}\big|\big|\boldsymbol{\varphi}(\boldsymbol{x}_j)-\boldsymbol{c}_+\big|\big|^2, \tag{36}$$

$$R_-^2 = \frac{1}{|I_R^+|}\sum_{i\in I_R^+}||\boldsymbol{\varphi}(\boldsymbol{x}_i)-\boldsymbol{c}_-||^2. \tag{37}$$

Once the centers $\boldsymbol{c}_\pm$ and square radiuses $R^2{}_\pm$ are known from (30), (34), (36), and (37), two hyperspheres

$$||\boldsymbol{\varphi}(\boldsymbol{x})-\boldsymbol{c}_+||^2\leq R_+^2\quad\text{and}\quad||\boldsymbol{\varphi}(\boldsymbol{x})-\boldsymbol{c}_-||^2\leq R_-^2 \tag{38}$$

are obtained. A new test sample $\boldsymbol{x}$ is assigned to class + or −, depending on which of the two hyperspheres given by (38) it lies closest to, i.e.,

$$f(\boldsymbol{x}) = \arg\min_{+,-}\left\{\frac{||\boldsymbol{\varphi}(\boldsymbol{x})-\boldsymbol{c}_+||^2}{R_+^2},\frac{||\boldsymbol{\varphi}(\boldsymbol{x})-\boldsymbol{c}_-||^2}{R_-^2}\right\}. \tag{39}$$

Compared the TSVM with TSVH for the nonlinear case, the former requires inversion of two matrices of size $(l+1)\times(l+1)$, respectively, along with two dual QPPs to be solved, whereas the latter only needs to solve two SVM-type dual QPPs without any matrix inversion in the

objective functions of its dual QPPs, making the TSVH surpass the TSVM in learning speed for large-scale problems. Further, this difference leads the learning algorithms of SVM can be easily extended to the TSVH, such as the SMO [13, 14] and geometric algorithms [18, 19].

In the next Section, we describe a learning algorithm for the TSVH based on the Gilbert's algorithm.

4 Learning algorithm for TSVH

In this Section, we describe a learning algorithm for the TSVH. Here, only (32) is considered because (33) is similar to (32).

Without the loss of generality, let us rewrite (32) with the matrix expression as follows:

$$\begin{aligned} \max \quad & -t_1\alpha^T\mathbf{K}_-\alpha + t_1\alpha^T\boldsymbol{u} \\ \text{s.t.} \quad & \alpha^T\boldsymbol{e}_2 = v_1,\ 0 \le \alpha \le C_1\boldsymbol{e}_2, \end{aligned} \tag{40}$$

where $\boldsymbol{e}_2$ is a vector of ones of l^- dimension, the kernel matrix $\mathbf{K}_- = (k_{j_1j_2}) = \left(k(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2})\right) \in \mathcal{R}^{l^-\times l^-}$, and $\boldsymbol{u} = (u_1; u_2; \ldots; u_{l^-})$ with $u_j = \sum_{i\in\mathcal{I}^+} k(\boldsymbol{x}_i, \boldsymbol{x}_j) - k(\boldsymbol{x}_j, \boldsymbol{x}_j)$, $j = 1, \ldots, l^-$. Note that t_1 is a constant (in general $t_1 > 0$ holds); therefore, the problem (40) can be rewritten as

$$\begin{aligned} \min \quad & \alpha^T\mathbf{K}_-\alpha - \alpha^T\boldsymbol{u} \\ \text{s.t.} \quad & \alpha^T\boldsymbol{e}_2 = v_1,\ 0 \le \alpha \le C_1\boldsymbol{e}_2. \end{aligned} \tag{41}$$

According to the constraint $\alpha^T\boldsymbol{e}_2 = v_1$, the linear term in the objective function of (41) can be expressed as

$$\begin{aligned} \alpha^T\boldsymbol{u} &= \left(\frac{1}{v_1}v_1\right)\alpha^T\boldsymbol{u} = \frac{1}{v_1}\left(\alpha^T\boldsymbol{u}v_1\right) = \frac{1}{v_1}\left(\alpha^T\boldsymbol{u}\boldsymbol{e}_2^T\alpha\right) \\ &= \frac{1}{2v_1}\left(\left(\alpha^T\boldsymbol{u}\boldsymbol{e}_2^T\alpha\right) + \left(\alpha^T\boldsymbol{e}_2\boldsymbol{u}^T\alpha\right)\right) \\ &= \frac{1}{2v_1}\alpha^T\left(\boldsymbol{u}\boldsymbol{e}_2^T + \boldsymbol{e}_2\boldsymbol{u}^T\right)\alpha. \end{aligned} \tag{42}$$

Substituting (42) into the objective function of (41) leads to

$$\begin{aligned} \alpha^T\mathbf{K}_-\alpha - \alpha^T\boldsymbol{u} &= \alpha^T\mathbf{K}_-\alpha - \frac{1}{2v_1}\alpha^T\left(\boldsymbol{u}\boldsymbol{e}_2^T + \boldsymbol{e}_2\boldsymbol{u}^T\right)\alpha \\ &= \alpha^T\left(\mathbf{K}_- - \frac{1}{2v_1}\left(\boldsymbol{u}\boldsymbol{e}_2^T + \boldsymbol{e}_2\boldsymbol{u}^T\right)\right)\alpha. \end{aligned}$$

Denote $\mathbf{G} = \mathbf{K}_- - \frac{1}{2v_1}\left(\boldsymbol{u}\boldsymbol{e}_2^T + \boldsymbol{e}_2\boldsymbol{u}^T\right) = (g_{j_1j_2})$, where

$$\begin{aligned} g_{j_1j_2} &= k_{j_1j_2} - \frac{1}{2v_1}(u_{j_1} + u_{j_2}) \\ &= k(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}) + \frac{1}{2v_1t_1}\left(k(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_1}) + k(\boldsymbol{x}_{j_2}, \boldsymbol{x}_{j_2})\right) \\ &\quad - \frac{1}{2v_1}\sum_{i\in\mathcal{I}^+}\left(k(\boldsymbol{x}_i, \boldsymbol{x}_{j_1}) + k(\boldsymbol{x}_i, \boldsymbol{x}_{j_2})\right), \end{aligned}$$

and set $\alpha_j := \alpha_j/v_1$, $j \in \mathcal{I}^-$, then the problem (41) is equivalent to the following QPP:

$$\begin{aligned} \min \quad & \alpha^T\mathbf{G}\alpha \\ \text{s.t.} \quad & \alpha^T\boldsymbol{e}_2 = 1,\ 0 \le \alpha \le \mu_1\boldsymbol{e}_2, \end{aligned} \tag{43}$$

where $\mu_1 = C_1/v_1$.

To optimize (43), let us first consider the problem of finding the minimum norm problem on the data set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, which can be expressed as

$$\begin{aligned} \min \quad & \left\|\sum_{k=1}^{n}\alpha_k\boldsymbol{x}_k\right\|^2 = \alpha^T\mathbf{M}\alpha \\ \text{s.t.} \quad & \alpha^T\boldsymbol{e} = 1,\ 0 \le \alpha \le \boldsymbol{e}, \end{aligned} \tag{44}$$

where $\boldsymbol{e}$ is a vector of ones of n dimension, and $\mathbf{M} = (m_{ij}) \in \mathcal{R}^{n\times n}$ with $m_{ij} = \boldsymbol{x}_i^T\boldsymbol{x}_j$, $i, j = 1, \ldots, n$. For this problem, Gilbert [39] proposed an iterative geometric algorithm for (44) to obtain the ε-optimal solution, called the Gilbert's algorithm, which is represented by the extreme points of the convex hull comprised of the points of X. This Gilbert's algorithm is described by the following three steps:

Algorithm 1: **Gilbert's algorithm**

1° *Initialization:* Set the initial point $\boldsymbol{x}$ as the any point of X, such as $\boldsymbol{x} = \boldsymbol{x}_1$;

2° *Stopping condition:* Find the point $\boldsymbol{x}_r$ such that $\boldsymbol{x}_r = \arg\min_{\boldsymbol{x}_k\in X}\{\langle\boldsymbol{x}_k, \boldsymbol{x}\rangle\}$, If the ε-optimality condition $\|\boldsymbol{x}\|^2 - \langle\boldsymbol{x}_r, \boldsymbol{x}\rangle < \varepsilon$ holds, then the point $\boldsymbol{x}$ is the ε-optimal solution. Otherwise, go to Step 3°;

3° *Adaptation*: Set $q = \frac{\langle\boldsymbol{x}, \boldsymbol{x}-\boldsymbol{x}_r\rangle}{\|\boldsymbol{x}-\boldsymbol{x}_r\|^2}$ if $\boldsymbol{x} - \boldsymbol{x}_r \neq 0$, otherwise $q = 0$; set $\boldsymbol{x} = \boldsymbol{x} + q(\boldsymbol{x}_r - \boldsymbol{x})$. Continue with Step 2°.

Obviously, (43) is similar to (44). Thus, we consider to solve (43) by using the Gilbert's algorithm. We first introduce the notion of reduced convex hull (RCH) [18, 40] for a point set, which is

Definition Let X be a set with n different points. For a real number $0 < \mu < 1$, the reduced convex hull (RCH) of X, denoted RCH(X, μ), is defined as

$$\mathrm{RCH}(X, \mu) = \left\{\boldsymbol{x}\ \middle|\ \boldsymbol{x} = \sum_{i=1}^{n}a_i\boldsymbol{x}_i, \boldsymbol{x}_i \in X, \sum_{i=1}^{n}a_i = 1, 0 \le a_i \le \mu\right\}. \tag{45}$$

Therefore, (43) can be regarded as the problem of finding the minimum norm problem on RCH(D^-,μ_1) with an *unknown distance*. Fortunately, it is not necessary to consider this distance for this problem since in the Gilbert's algorithm only the inner product $\langle\cdot,\cdot\rangle$ is used. For briefly,

we use $\langle \boldsymbol{u}_{j_1}, \boldsymbol{u}_{j_2} \rangle_{\mathbf{G}}$ to denote the inner product of $\boldsymbol{\varphi}(\boldsymbol{x}_{j_1})$ and $\boldsymbol{\varphi}(\boldsymbol{x}_{j_2})$ with this unknown distance $||\cdot||_{\mathbf{G}}$, i.e., $g_{j_1 j_2} = \langle \boldsymbol{u}_{j_1}, \boldsymbol{u}_{j_2} \rangle_{\mathbf{G}}$, where $\boldsymbol{u} = \boldsymbol{\varphi}(\boldsymbol{x})$. Mavroforakis and Theodoridis [18] pointed out that the extreme points of RCH(X, μ) are the combinations of points in X.

Proposition The extreme point of the RCH(X, μ) has coefficient a_i belonging to the set $S = \{0, \mu, 1 - \lfloor 1/\mu \rfloor \mu\}$, that is, each extreme point of RCH(X, μ) is a convex combination of $m = \lceil 1/\mu \rceil$ different points in X. Furthermore, if $1/\mu = \lceil 1/\mu \rceil$, then all $a_i = \mu$; otherwise, $a_i = \mu$, $i = 1, \ldots, m-1$, and $a_m = 1 - \lfloor 1/\mu \rfloor \mu$.

This proposition provides the necessary but not sufficient condition for determining the extreme points of RCH, in which the number of points satisfying the condition is obviously larger than that of extreme points. Therefore, it is impossible to inspect all candidate points in the process of finding the closest point. In fact, it need not consider all possible extreme points of RCH in the geometric algorithm, but only compute the projections of all negative points in $\mathcal{I}^-$, and choose the first $\lceil 1/\mu_1 \rceil$ points with the smallest projections to update the current point.

Based on the above discussion on RCH, we describe the learning algorithm for TSVH based on the Gilbert's one as follows.

Algorithm 2: **Learning algorithm for TSVH**

1° *Initialization:* Set the parameters C_1, ν_1 and kernel; set $\mu_1 = \min\{1, C_1/\nu_1\}$, $\lambda_1 = 1 - \lfloor 1/\mu_1 \rfloor$, and $m_1 = \lceil 1/\mu_1 \rceil$; set the initial point $\boldsymbol{u}$ as the centroid point of RCH(D^-, μ_1), i.e., set $\alpha_j = 1/|\mathcal{I}^-|$, $j \in \mathcal{I}^-$.

2° *Stopping condition:* Find the point $\boldsymbol{u}_r^{opt} = \arg\min_r \{\langle \boldsymbol{u}_r, \boldsymbol{u} \rangle_{\mathbf{G}}\}$, where $\boldsymbol{u}_r = \sum_{j \in \mathcal{I}^-} b_j \boldsymbol{u}_j$, $b_j \in \{0, \lambda_1, \mu_1\}$, $\sum_{j \in \mathcal{I}^-} b_j = 1$. If the ε-optimality condition $||\boldsymbol{u}||_{\mathbf{G}}^2 - \langle \boldsymbol{u}_r^{opt}, \boldsymbol{u} \rangle_{\mathbf{G}} < \varepsilon$ holds, then the point $\boldsymbol{u}$ is the ε-optimal solution. Otherwise, go to Step 3°.

3° *Adaptation*: Set $q = \frac{\langle \boldsymbol{u}, \boldsymbol{u} - \boldsymbol{u}_r^{opt} \rangle_{\mathbf{G}}}{||\boldsymbol{u} - \boldsymbol{u}_r^{opt}||_{\mathbf{G}}^2}$ if $\boldsymbol{u} - \boldsymbol{u}_r^{opt} \neq 0$, otherwise $q = 0$; update $\boldsymbol{u} = \boldsymbol{u} + q(\boldsymbol{u}_r^{opt} - \boldsymbol{u})$,i.e., set $\alpha_j = \alpha_j + q(b_j - \alpha_j)$, $j \in \mathcal{I}^-$. Continue with Step 2°.

For this algorithm, it has almost the same complexity as the Gilbert's algorithm; the same caching scheme can be used, with only $\mathcal{O}(l^{\pm})$ storage requirements.

5 Experiments

To validate the effectiveness of our proposed method, we compare the performances of TSVH, TSVM, and SVM on some data sets. All these algorithms are implemented in Matlab 7.8 [41] on Windows XP running on a PC with system configuration Intel(R) Core(TM)2 Duo CPU (2.26 GHz) with 2 GB of RAM. We simulate them on several artificial and UCI benchmark data sets[2] which are commonly used in testing machine learning algorithms. Note that we only consider the Gaussian kernel in the experiments. To fairly compare these algorithms, we demonstrate the performances of these algorithms using the accuracy and CPU time for learning. Another important problem is the parameter selection for these algorithms. To fairly reflect the performances of these algorithms, we select the parameters C's and ν's from the set of values $\{2^i \cdot l^{\pm} | i = -9, -8, \ldots, 10\}$ and select the kernel parameters γ's from the set of values $\{2^i | i = -9, -8, \ldots, 10\}$ by tuning a set comprising of about random 10–30 percent of the sample set in the simulations. For the learning of the SVM and TSVM, we employ the geometric methods to learn their separating hyperplanes, see Refs. [18, 19, 27]. In addition, the kernel matrices (or the inversions of kernel matrices) of these algorithms are cached in memory before learning.

5.1 Toy examples

In order to illustrate graphically the effectiveness of our TSVH, we first test its ability to learn the artificially generated Ripley's synthetic data set [42] in two dimensions. In this Ripley's data set, both class "+" and class "−" are generated from mixtures of two Gaussians with the classes overlapping to the extent that the Bayes accuracy is around 92 %.

Figure 1 shows the classification results of SVM, TSVM, and TSVH with Gaussian kernels on this Ripley's data set. It can be seen that the hyperspheres of TSVH effectively cover the different classes of samples and the corresponding separating hyperplane obtains the best classification result. While the nonparallel hyperplanes of TSVM do not effectively reflect the distributions of two classes of samples, they only cross as many as possible samples of the corresponding classes. Table 1 also lists the test accuracies of these algorithms. It can be seen that our TSVH obtains the best accuracy (about 92.1 %) among these methods, which is close to the Bayes classification result. Whereas the TSVM obtains the worst test accuracy (about 90.2 %) among these methods, which effectively confirms the analysis on TSVM, that its nonparallel hyperplanes do not effectively describe the characterizations of two classes of samples. A more possible reason for the good performance of TSVH is that the hyperspheres of TSVH are more effective to depict the class information of data.

Another artificial example is the checkerboard problem. For the checkerboard problem, it consists of a series of uniform points in $\mathcal{R}^2$ of red and blue points taken from

[2] Available at: http://archive.ics.uci.edu/ml/.

4×4 red and blue squares of a checkerboard. This is a tricky test case in data mining algorithms for testing performances of nonlinear classifiers. In the simulations, we consider the one non-overlapping and two overlapping cases for the checkerboard problem, that is, three sets of 800 (Class +: 400, Class −: 400) randomly generated training points on a 2-dimensional checkerboard of 4×4 cells are used. Each sample attribute ranges from 0 to 4 and the margins are 0, −0.05, and −0.1, respectively (the negative value indicating the overlapping between classes, that is, the overlapping of the cells), and the 3200 test samples are randomly generated with margin 0.

Figures 2 and 3 show the simulation results of these three algorithms on the first two training set because of the limitation of paper. It can be seen that our TSVH still obtains the best classification results, while TSVM obtains the worst results among these methods, especially for the overlapping case. Formally, the two hyperspheres of TSVH effectively cover the corresponding classes of samples, while the hyperplanes of TSVM do not reflect the information of samples. For the latter, they still pass through as many as possible samples of the corresponding classes, leading TSVM to obtain the worst separating hyperplane. However, we can obtain a pair of more smooth hyperplanes for the TSVM if we add the terms $\epsilon_i(\|\boldsymbol{w}_\pm\|^2 + b_\pm^2)/2, \lambda_i > 0, i = 1, 2$ into the objective functions of the TSVM. Table 1 also gives the test accuracies and learning

Table 1 Performance comparisons on Ripley and checkerboard data sets

Dataset	SVM	TSVM	TSVH
Ripley			
Accuracy (%)	90.6	90.2	92.1
Time (s)	0.087	0.029	0.021
Checkerboard			
$m_a = 0$[a]			
Accuracy	95.56	95.52	96.69
Time	26.45	6.77	6.51
Checkerboard			
$m_a = -0.05$			
Accuracy	93.44	92.94	96.56
Time	27.52	6.85	6.71
Checkerboard			
$m_a = -0.10$			
Accuracy	91.19	87.19	96.13
Time	28.12	6.99	6.92

[a] m_a is the margin

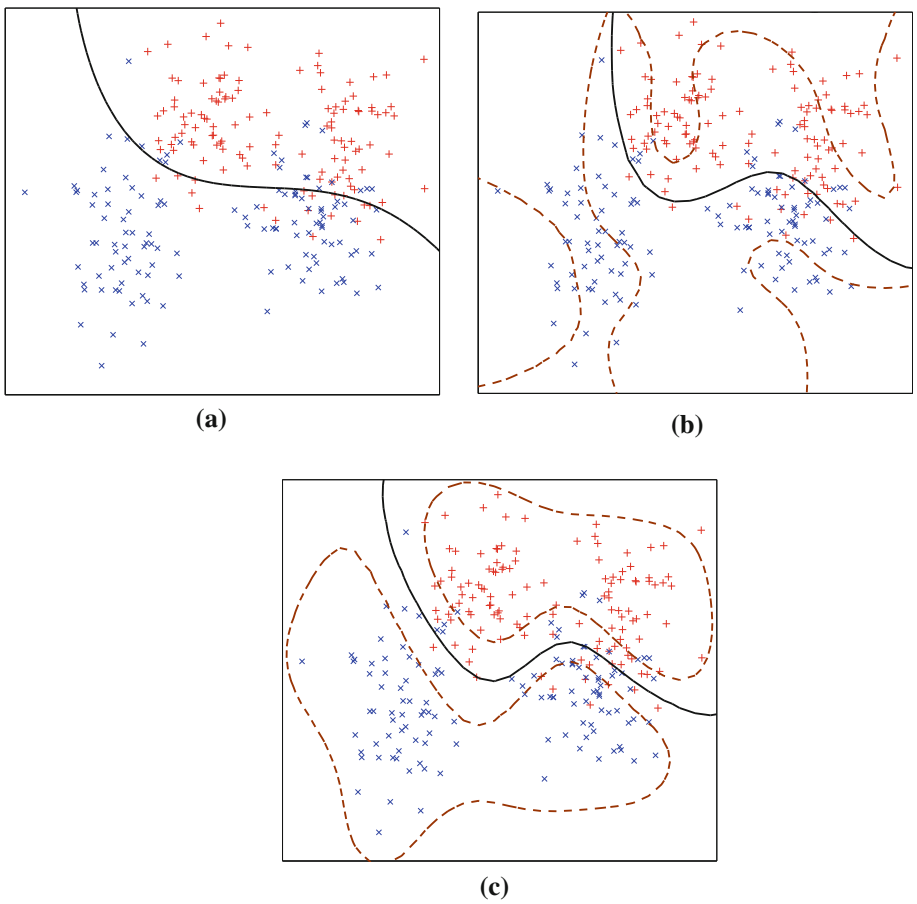

Fig. 1 Classification results of SVM (**a**), TSVM (**b**), and TSVH (**c**) on Ripley's data set. The two classes of training samples are symbolled by " × " and "+," the separating hyperplanes are shown with *solid curves*, and the nonparallel hyperplanes/hyperspheres of TSVM/TSVH are shown with *dashed curves*

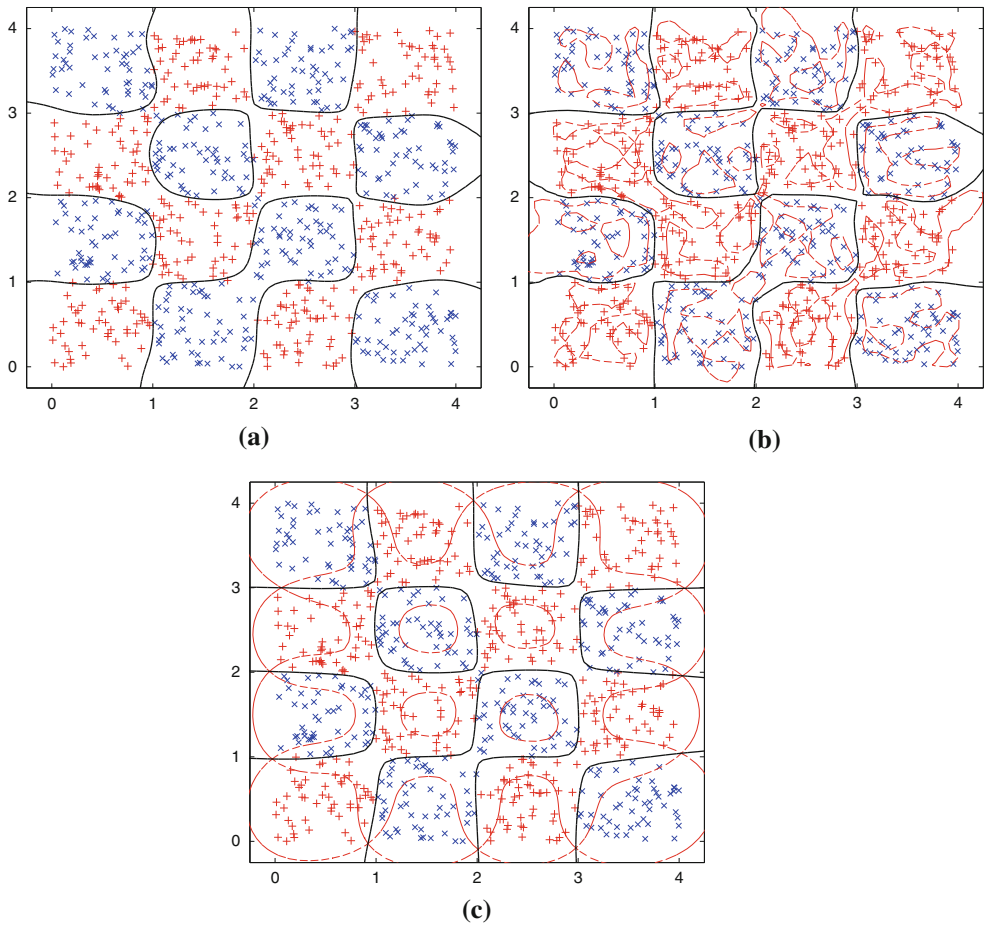

Fig. 2 Classification results of SVM (**a**), TSVM (**b**), and TSVH (**c**) with Gaussian kernel on checkerboard problem with margin 0. The separating hyperplanes are shown with *solid curves*, and the nonparallel hyperplanes/hyperspheres of TSVM/TSVH are shown with *dashed curves*

CPU time of these methods on these cases with 30 independent runs, which confirm effectively our results. As for the learning CPU time, Table 1 shows that the TSVH and TSVM are about four times faster than the SVM, indicating the learning speed of the TSVH is much faster than the SVM, whereas the learning speed of the TSVH is slightly faster than the TSVM. However, if we consider the extra time for matrix inversions in the TSVM, the learning time of the TSVM will be obviously larger than the TSVH. In addition, we should point that the TSVH introduces an extra parameter compared with the TSVM, which means the total CPU time required for cross-validation may be much larger than the TSVM and SVM. Hence, an important further work is to discuss a suitable method for determining the parameters of TSVH.

5.2 Benchmark data sets

To further test the performance of TSVH, we run these models on several publicly available benchmark data sets, which are commonly used in test machine learning algorithms, and investigate the results in terms of accuracy and training CPU time. We use the one-vs-one method to deal with the multi-classification data sets and take the average CPU time as the learning time. Table 2 lists the descriptions of these benchmark data sets. Note that we use the tenfold cross-validation method to simulate the data sets in Table 2.

Table 3 lists the average learning results of SVM, TSVM, and TSVH on these benchmark data sets with tenfold cross-validation run, including the accuracies and learning CPU time. It can be seen that the TSVH obtains the comparable generalization with the SVM and TSVM. As for the learning CPU time, it can be found that both methods are about four times faster than that of the SVM. This is because they solve two smaller sized QPPs instead of a single QPP in the classical SVM. Besides, as the artificial examples, the learning time of the TSVH is similar to that of the TSVM for the same reason. However, the CPU time for the model selection of the TSVH is larger

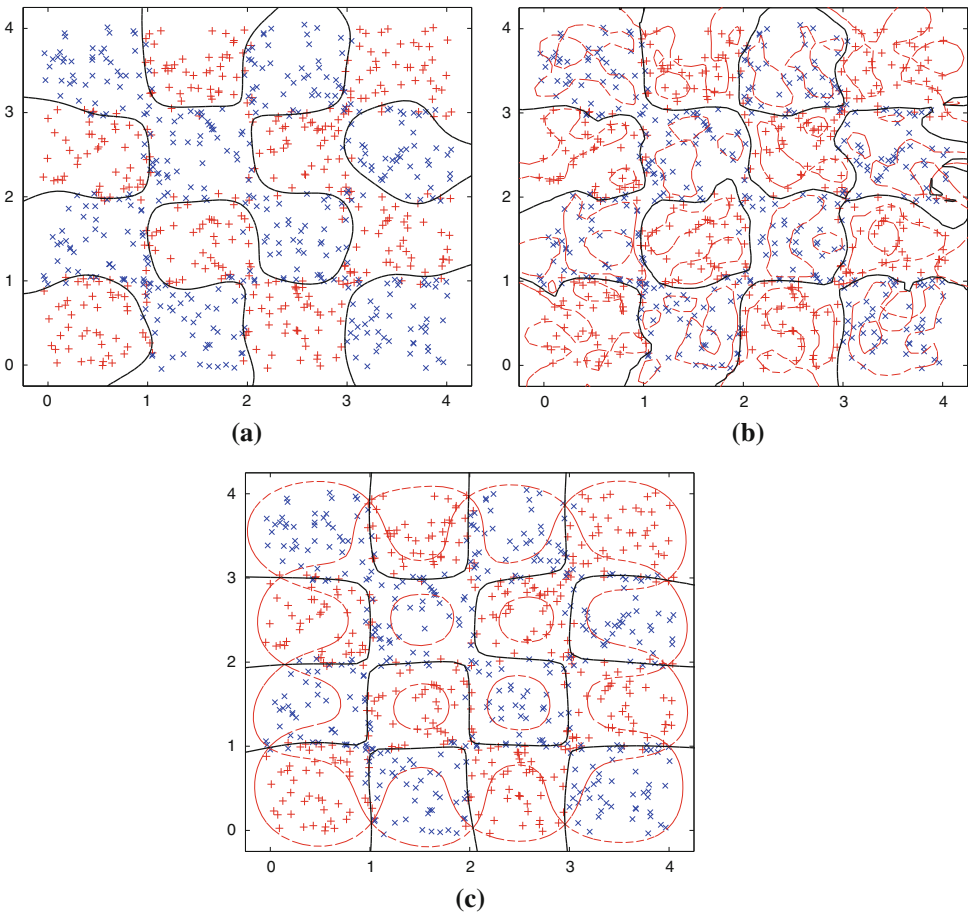

Fig. 3 Classification results of SVM (**a**), TSVM (**b**), and TSVH (**c**) with Gaussian kernel on checkerboard problem with margin -0.05. The separating hyperplanes are shown with *solid curves*, and the nonparallel hyperplanes/hyperspheres of TSVM/TSVH are shown with *dashed curves*

Table 2 Descriptions of the UCI benchmark data sets

Dataset	No. samples	Attribute	Class
German	1,000	20	2
Harberman	306	3	2
Heart	270	13	2
Iris	150	4	3
Optdigits	5,620	64	10
Pendigits	10,992	16	10
Ringnorm	400	20	2
Segemention	2,210	19	7
Splice	3,175	60	2
Thyroid	215	5	2
Titanic	2,201	3	2
Twonorm	400	20	2
USPS	9,298	256	10
Waveform	5,000	21	3

than the TSVM and SVM since an extra penalty factor is introduced in it.

We now use the Wilcoxon signed-ranks test [43], which is a simple, yet safe and robust nonparametric tests for statistical comparisons of classifiers, to compare the generalization performance of the TSVH, TSVM, and SVM. Let d_i be the difference between the test accuracies of the two classifiers on ith out of N data sets. The differences are ranked according to their absolute values. Let r^+ be the sum of ranks for the data sets on which the second algorithm outperformed the first, and r^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored. Let T be the smaller of the sums, $T = \min(r^+, r^-)$. Then, the statistics

$$Z = \frac{T - N(N+1)/4}{\sqrt{N(N+1)(2N+1)/24}}$$

is distributed approximately normally.

Table 4 shows the comparisons on the benchmark data sets between the results of TSVH and TSVM, TSVH and SVM, and TSVM and SVM, respectively. It can be seen that, given $\alpha = 0.05$ (then, $Z_{\alpha/2} = -1.96$), our TSVH derives the obviously better generalization than the TSVM and SVM, while the TSVM obtains the comparable generalization with the SVM. Therefore, our TSVH is the better machine learning algorithm for classifications than the SVM and TSVM. This confirms that the hyperspheres of TSVH can more effectively describe the

Table 3 Performance comparisons on the UCI benchmark data sets

Dataset	SVM		TSVM		TSVH	
	Accuracy	Time	Accuracy	Time	Accuracy	Time
German	75.13 ± 2.67	39.76	75.49 ± 2.79	10.96	75.48 ± 2.61	10.35
Harberman	73.72 ± 1.97	0.50	73.12 ± 2.31	0.15	74.21 ± 1.64	0.13
Heart	80.22 ± 4.31	1.12	80.25 ± 4.22	0.30	80.31 ± 3.41	0.29
Iris	99.00 ± 1.61	0.017	98.67 ± 1.72	0.0041	99.00 ± 1.61	0.0025
Optdigits	98.37 ± 1.74	115.70	98.55 ± 1.69	29.84	98.94 ± 1.53	27.35
Pendigits	99.21 ± 1.91	470.33	98.82 ± 2.72	118.47	99.26 ± 1.29	112.31
Ringnorm	96.43 ± 0.61	8.19	96.31 ± 0.73	2.12	96.62 ± 0.64	2.01
Segemention	98.31 ± 3.16	52.50	98.35 ± 3.41	13.62	98.46 ± 3.32	12.34
Splice	89.83 ± 1.64	843.41	89.78 ± 1.95	214.62	89.82 ± 1.79	207.25
Thyroid	96.14 ± 2.93	0.65	96.19 ± 2.71	0.16	96.11 ± 2.68	0.15
Titanic	77.43 ± 6.67	433.83	77.15 ± 6.79	109.45	77.36 ± 6.31	107.63
Twonorm	96.69 ± 0.61	8.37	96.47 ± 0.59	2.18	96.82 ± 0.57	2.06
USPS	98.51 ± 1.45	517.53	98.69 ± 1.53	136.63	98.58 ± 1.72	133.91
Waveform	92.31 ± 1.41	658.41	92.60 ± 1.65	148.46	92.48 ± 1.57	144.52
Average	90.81		90.75		90.96	

Table 4 Statistics comparisons of accuracies of TSVH, TSVM, and SVM

Dataset	TSVH versus TSVM		TSVH versus SVM		TSVM versus SVM	
	Difference	Rank	Difference	Rank	Difference	Rank
German	−0.01	1	0.35	12	0.36	12
Harberman	1.09	14	0.49	13	−0.60	14
Heart	0.06	3	0.09	7	0.03	1
Iris	0.33	10	0	1	−0.33	11
Optdigits	0.49	13	0.57	14	0.18	6.5
Pendigits	0.44	12	0.05	4	−0.39	13
Ringnorm	0.31	9	0.19	11	0.12	5
Segemention	0.11	5.5	0.15	9	0.04	2
Splice	0.04	2	−0.01	2	−0.05	3.5
Thyroid	−0.08	4	−0.03	3	0.05	3.5
Titanic	0.21	8	−0.07	5.5	−0.28	9
Twonorm	0.35	11	0.13	8	−0.22	8
USPS	−0.11	5.5	0.07	5.5	0.18	6.5
Waveform	−0.12	7	0.17	10	0.29	10
Z	−2.20		−2.64		−0.69	

data information than the nonparallel hyperplanes of TSVM.

6 Conclusions

In this paper, we have proposed a novel SVM approach to data classification, termed the TSVH. The introduction of TSVH is motivated by the SVDD, a special case of the OCSVM, and the TSVM, that is, in the TSVH, we solve two QPPs of smaller sizes instead of a large sized one as we do in the classical SVM, each QPP is based on the SVDD, which requires the samples of one class to be covered by a hypersphere as many as possible, while the samples of the other class to be as possible as far from this hypersphere. The strategy that solving two smaller sized QPPs, roughly of size (l/2), makes the TSVH be almost four times faster than a classical SVM classifier. Compared with the TSVM,

the TSVH has some obvious merits. First, it does not need to find the inversion matrices in its pair dual QPPs, while the TSVM needs the inversions of two matrices of size $(l + 1) \times (l + 1)$. This leads the TSVH to be more suitable large-scale problems by combining efficient SVM algorithms. Second, the TSVH uses two hyperspheres but not hyperplanes to describe the samples of two classes, which leads the TSVH to be more suitable for general cases; for instance, the hyperplanes in the TSVM cannot effectively describe the samples in Ripley's data set. Computational comparisons with the TSVM and SVM in terms of classification accuracy and learning time have shown that the proposed TSVH is not only faster than the other two methods, but also obtains comparable generalization.

The further work is to validate the performance of TSVH in more real classification problems, such as face recognition, image classification, and text categorization. Another important further work is to discuss the theoretical illustration for TSVH since TSVH is an indirect classifier. The third important further work is to find an efficient prediction algorithm for TSVH in further because of the loss of sparsity in TSVH. Last, notice that TSVH introduces more parameters, which makes the efficiency of model selection for TSVH be low; thus, it is necessary to find some efficient method for determining the parameters in TSVH.

Acknowledgments The authors would like to thank the anonymous reviewers for their constructive comments and suggestions. This work is supported by the National Natural Science Foundation of China (61202156), the National Natural Science Foundation of Shanghai (12ZR1447100), the Innovative Project of Shanghai Municipal Education Commission (11YZ81), and the program of Shanghai Normal University (DZL121, SK201204).

References

1. Burges CJC (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Discov 2(2):121–167
2. Christianini V, Shawe-Taylor J (2002) An introduction to support vector machines. Cambridge University Press, Cambridge
3. Vapnik VN (1995) The natural of statistical learning theory. Springer, New York
4. Vapnik VN (1998) Statistical learning theory. Wiley, New York
5. Lee S, Verri A (2002) Pattern recognition with support vector machines. In: First international workshop, SVM 2002, Springer, Niagara Falls
6. Osuna E, Freund R, Girosi F (1997) Training support vector machines: an application to face detection. In: Proceedings of IEEE computer vision and pattern recognition, San Juan, Puerto Rico, pp 130–136
7. Joachims T, Ndellec C, Rouveriol C (1998) Text categorization with support vector machines: learning with many relevant features, In: European conference on machine learning No.10, Chemnitz, Germany, pp 137–142
8. Brown MPS, Grundy WN, Lin D, et al (2000) Knowledge-based analysis of microarray gene expression data by using support vector machine. Proc Natl Acad Sci USA 97(1):262–267
9. Ebrahimi T, Garcia GN, Vesin JM (2003) Joint time-frequency-space classification of EEG in a brain–computer interface application. J Apply Signal Process 1(7):713–729
10. Huang Z, Chen H, Hsua C-J, Chen W-H, Wu S (2004) Credit rating analysis with support vector machines and neural networks: a market comparative study. Decis Support Syst 37:543–558
11. Cortes C, Vapnik VN (1995) Support vector networks. Mach Learn 20:273–297
12. Osuna E, Freund R, Girosi F (1997) Support vector machines: training and applications. Technical report AIM1602, MIT Artificial Intelligence Laboratory, Cambridge, MA
13. Platt J (1999) Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges CJC, Smola AJ (eds) Advances in Kernel methods–support vector learning. MIT Press, Cambridge, MA, pp 185–208
14. Keerthi SS, Shevade SK, Bhattacharyya C, Murthy KRK (2001) Improvements to platt's SMO algorithm for SVM classifier design. Neural Comput 13(3):637–649
15. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
16. Suykens JAK, Lukas L, van Dooren P, De Moor B, Vandewalle J (1999) Least squares support vector machine classifiers: a large scale algorithm. In: Proceedings of European conference of circuit theory design, pp 839–842
17. Bennett KP, Bredensteiner EJ (2000) Duality and geometry in SVM classifiers. In: Langley P (ed) Proceedings of the 17th international conference on machine learning. Morgan Kaufmann, Los Altos, CA, pp 57–64
18. Mavroforakis ME, Theodoridis S (2006) A geometric approach to support vector machine (SVM) classification. IEEE Trans Neural Netw 17(3):671–682
19. Tao Q, Wu G, Wang J (2008) A general soft method for learning SVM classifiers with L_1-norm penalty. Pattern Recogn 41(3): 939–948
20. Wang J, Tao Q, Wang J (2002) Kernel projection algorithm for large-scale SVM problems. J Comput Sci Tech 17(5):556–564
21. Mangasarian OL, Wild EW (2006) Multisurface proximal support vector classification via generalized eigenvalues. IEEE Trans Pattern Anal Mach Intell 28(1):69–74
22. Jayadeva, Khemchandani R, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910
23. Kumar MA, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36(4): 7535–7543
24. Kumar MA, Gopal M (2008) Application of smoothing technique on twin support vector machines. Pattern Recogn Lett 29(13):1842–1848
25. Ghorai S, Dutta PK, Mukherjee A (2010) Newton's method for nonparallel plane proximal classifier with unity norm hyperplanes. Signal Process 90(1):93–104
26. Ghorai S, Mukherjee A, Dutta PK (2009) Nonparallel plane proximal classifier. Signal Process 89(4):510–522
27. Peng X (2010) A ν-twin support vector machine (ν-TSVM) classifier and its geometric approaches. Inf Sci 180(15):3863–3875
28. Ye Q, Zhao C, Ye N, Chen X (2011) Localized twin SVM via convex minimization. Neurocomputing 74:580–587
29. Peng X (2010) TSVR: an efficient twin support vector machine for regression. Neural Netw 23(3):365–372
30. Peng X (2010) Primal twin support vector regression and its sparse approximation. Neurocomputing 73(16-18):2846–2858
31. Peng X (2011) TPMSVM: A novel twin parametric-margin support vector machine for pattern recognition. Pattern Recogn 44(10–11):2678–2692
32. Peng X (2012) Efficient twin parametric insensitive support vector regression model. Neurocomputing 79:26–38

33. Tax D, Duin R (1999) Support vector domain description. Pattern Recogn Lett 20:1191–1199
34. Tax D, Duin R (2004) Support vector data description. Mach Learn 54:45–66
35. Schölkopf B, Platt J, Shawe-Taylor J, Smola AJ, Williamson RC (2001) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471
36. Mercer J (1909) Functions of positive and negative type and the connection with the theory of integal equations. Philos Trans R Soc Lond Ser A 209:415–446
37. Peng X (2011) Building sparse twin support vector machine classifiers in primal space. Inf Sci 181(18):3967–3980
38. Shao Y, Zhang C, Wang X, Deng N (2011) Improvements on twin support vector machines. IEEE Trans Neural Netw 22(6): 962–968
39. Gilbert EG (1966) An iterative procedure for computing the minimum of a quadratic form on a convex set. SIAM J Control 4(1):61–79
40. Crisp DJ, Burges CJC (2000) A geometric interpretation of ν-SVM classifiers. In: Solla S, Leen T, Muller K-R (eds) Advances in neural information processing systems, vol 12, pp 244–250
41. MATLAB, User's guide, the MathWorks, Inc., 1994–2001, http://www.mathworks.com
42. Ripley BD (1996) Pattern recognition and neural networks. Cambridge University Press, Cambridge
43. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30