Information Sciences 180 (2010) 3863–3875

Contents lists available at ScienceDirect

Information Sciences

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

# A $v$-twin support vector machine ($v$-TSVM) classifier and its geometric algorithms

Peng Xinjun *

*Department of Mathematics, Shanghai Normal University, Shanghai 200234, PR China*
*Scientific Computing Key Laboratory of Shanghai Universities, Shanghai 200234, PR China*

ARTICLE INFO

*Article history:*
Received 27 February 2009
Received in revised form 26 June 2010
Accepted 28 June 2010

*Keywords:*
Pattern recognition
Twin support vector machine
Geometric interpretation
Geometric algorithm
Probabilistic speed-up strategy

ABSTRACT

In this paper, a $v$-twin support vector machine ($v$-TSVM) is presented, improving upon the recently proposed twin support vector machine (TSVM). This $v$-TSVM introduces a pair of parameters ($v$) to control the bounds of the fractions of the support vectors and the error margins. The theoretical analysis shows that this $v$-TSVM can be interpreted as a pair of minimum generalized Mahalanobis-norm problems on two reduced convex hulls (RCHs). Based on the well-known Gilbert's algorithm, a geometric algorithm for TSVM (GA-TSVM) and its probabilistic speed-up version, named PGA-TSVM, are presented. Computational results on several synthetic as well as benchmark datasets demonstrate the significant advantages of the proposed algorithms in terms of both computation complexity and classification accuracy.



## 1. Introduction

The support vector machine (SVM), introduced by Vapnik and co-workers [4,34,35], is an excellent tool for binary data classification. Compared with other machine-learning approaches, e.g., artificial neural networks (ANN) [28], the SVM possesses many advantages. First, the SVM solves a quadratic programming problem (QPP), which assures that once a solution has been reached, it is the unique (global) solution. Second, the SVM derives its sparse and robust solution by maximizing the margin between the two classes. Third, the SVM implements the structural risk minimization principle [34,35], which minimizes the upper bound of the generalization error, but not the empirical risk minimization principle. Fourth, it has intuitively geometric interpretations of classification tasks [1,15,37]. Recently, the SVM has been successfully applied in many fields [2,3,7,13,14,22,23,27].

One of the main challenges for the classical SVM is the high computational complexity of the quadratic programming problem (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. Recently, a major research topic related to the SVM has been the development of efficient learning algorithms and models. Over the past few decades, many improvements to the SVM have emerged, such as the chunking algorithm [5], the decomposition method [24], sequential minimal optimization (SMO) [16,26], the least squares support vector machine (LS-SVM) [30,31], the generalized eigenvalue proximal SVM (GEPSVM) [19], and geometric algorithms [1,8,20,32,33,36].

* Address: Department of Mathematics, Shanghai Normal University, Shanghai 200234, PR China. Tel.: +86 21 64324866.
*E-mail address:* xjpeng@shnu.edu.cn

0020-0255/$ - see front matter Crown Copyright © 2010 Published by Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.06.039

Recently, Jayadeva et al. [12] proposed a twin support vector machine (TSVM) classifier for binary classification, motivated by GEPSVM [19]. TSVM determines two nonparallel hyperplanes by two smaller and related SVM-type problems, in which each hyperplane is closer to one of the two classes and is as far as possible from the other one. The strategy of solving two smaller QPPs rather than a single large QPP makes the learning speed of TSVM approximately 4 times faster than that of a classical SVM. Some extensions to the TSVM include the least squares version of the TSVM (LS-TSVM) [18], a smooth TSVM [17], a nonparallel-plane proximal classifier (NPPC) [9,10], and twin support vector regression (TSVR) [25].

The experimental results in [12] have shown that the TSVM compares favorably with the SVM and the GEPSVM. However, in contrast to the classical SVM, the TSVM only aims at minimizing the empirical risks of training examples. For instance, to find the positive hyperplane, it minimizes simultaneously the $L_2$-norm empirical risks of positive samples and the $L_1$-norm empirical risks of negative samples with a trade-off parameter. Specifically, the corresponding QPP not only minimizes the distances from the positive hyperplane to the positive training samples but also requires the distances from the positive hyperplane to the negative samples to be at least one; otherwise, slack variables are added to avoid contradictions. This principle may lead to an increase in the number of support vectors (SVs) and reduce the generalization performance of the TSVM.

In this paper, we present a new version of the TSVM, named the $v$-twin support vector machine ($v$-TSVM), by introducing additional variables $\rho$ and new parameters $v$. Similar to the $v$-support vector machine ($v$-SVM) [29], the parameters $v$ in the $v$-TSVM have a better theoretical interpretation than the penalty factors $C$'s in the TSVM, which control the bounds of fractions of the numbers of SVs and the margin errors. We also show that the proposed $v$-TSVM can be interpreted as a pair of minimum generalized Mahalanobis-norm problems on two reduced convex hulls (RCHs) [6] composed of the positive and negative training samples, respectively. In the spirit of the well-known Gilbert's algorithm [11], we next propose a geometric algorithm for the TSVM (GA-TSVM) and its probabilistic speed-up version by combining it with the novel probabilistic speed-up strategy and sorting method, yielding the probabilistic speed-up GA-TSVM (PGA-TSVM), which performs fewer kernel evaluations than the GA-TSVM. Computational comparisons on several artificial and benchmark datasets show that the proposed algorithms derive better generalizations with fewer kernel evaluations and numbers of SVs, as well as less training CPU time than the other algorithms.

The remainder of this paper is organized as follows. In Section 2, we briefly review the SVM and TSVM. In Section 3, we discuss the $v$-TSVM in detail. Section 4 shows the geometric interpretation of the $v$-TSVM. Section 5 presents the geometric algorithms for the TSVM. Experiments on several artificial and benchmark datasets are presented in Section 6, and further possible improvements are also indicated. Finally, Section 7 summarizes and concludes the paper.

## 2. Background

Consider a classification problem with the dataset $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}$, where $\boldsymbol{x}_i \in \mathcal{X} \subset \mathcal{R}^m$ and $y_i \in \{-1, 1\}$. Denote by $I^{\pm}$ the sets of indices $i$ such that $y_i = \pm 1$, and by $I$ the set of all indices, i.e., $I = I^+ \bigcup I^-$. In the following, we also use $I^+$ and $I^-$ to represent the sets of positive and negative training examples.

### 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 some feature space $\mathcal{H}$

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

maximizing the total (interclass) margin $2/(\boldsymbol{w}^T\boldsymbol{w})$, and satisfying

$$y_i(\boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}_i) + b) \geqslant 1, \quad i \in I, \tag{2}$$

where $\boldsymbol{\varphi}(\cdot) : \mathcal{X} \to \mathcal{H}$ maps $\mathcal{X}$ into $\mathcal{H}$ and $\boldsymbol{w} \in \mathcal{H}$. For the linear case, we have $\boldsymbol{\varphi}(\boldsymbol{x}) = \boldsymbol{x}$. Under the Mercer theorem [34,35,21], 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 $k(\boldsymbol{u}, \boldsymbol{v}) = \exp\{-\gamma \|\boldsymbol{u} - \boldsymbol{v}\|_2^2\}$ with $\gamma > 0$, where $\|\cdot\|_2$ is the $L_2$-norm. To fit the practical application, one method is to add a slack variable $\xi_i$ for each $\boldsymbol{x}_i$, which allows a controlled violation of the constraint. Therefore, the SVM can be expressed as the following QPP with linear inequalities in $\mathcal{H}$:

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i \in I} \xi_i, \\ \text{s.t.} \quad & y_i\left(\boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}_i) + b\right) \geqslant 1 - \xi_i, \\ & \xi_i \geqslant 0, \quad i \in I, \end{aligned} \tag{3}$$

where $C > 0$ is the pre-specified regularization factor. By introducing the Lagrangian coefficients $\alpha_i$, we derive its dual QPP as follows:

$$\begin{aligned} \max \quad & \sum_{i\in I}\alpha_i - \frac{1}{2}\sum_{i,j\in I}\alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i,\boldsymbol{x}_j), \\ \text{s.t.} \quad & 0 \leqslant \alpha_i \leqslant C, \quad i \in I, \\ & \sum_{i\in I}\alpha_i y_i = 0. \end{aligned} \tag{4}$$

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

$$f(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i\in I}\alpha_i y_i k(\boldsymbol{x}_i,\boldsymbol{x}) + b\right), \tag{5}$$

where $\boldsymbol{w}$ is derived by the Karush–Kuhn–Tucker (KKT) optimality condition:

$$\boldsymbol{w} = \sum_{i\in I} y_i\alpha_i\boldsymbol{\varphi}(\boldsymbol{x}_i) = \sum_{i\in I^+}\alpha_i\boldsymbol{\varphi}(\boldsymbol{x}_i) - \sum_{j\in I^-}\alpha_j\boldsymbol{\varphi}(\boldsymbol{x}_j). \tag{6}$$

Note that optimizing the dual QPP (4) is time-consuming for large-scale problems because of the enormous matrix sizes and CPU-intensive matrix operations. Thus, a major research topic related to the SVM is to focus on fast learning (see Refs. [1,5,8,16,19,20,24,26,32,33,36]).

### 2.2. Twin support vector machine

The TSVM is a binary classifier that performs classification using two nonparallel hyperplanes instead of the single hyperplane used in the classical SVM [12]. These two hyperplanes 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 $\mathcal{R}^m$:

$$\boldsymbol{w}_+^T\boldsymbol{x} + b_+ = 0, \quad \boldsymbol{w}_-^T\boldsymbol{x} + b_- = 0, \tag{7}$$

such that each hyperplane is closer to the examples of one of the two classes 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 on 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:

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i\in I^+}\left(\boldsymbol{w}_+^T\boldsymbol{x}_i + b_+\right)^2 + C_1\sum_{j\in I^-}\xi_j, \\ \text{s.t.} \quad & -\boldsymbol{w}_+^T\boldsymbol{x}_j - b_+ \geqslant 1 - \xi_j, \\ & \xi_j \geqslant 0, \quad j \in I^-, \end{aligned} \tag{8}$$

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{j\in I^-}\left(\boldsymbol{w}_-^T\boldsymbol{x}_j + b_-\right)^2 + C_2\sum_{i\in I^+}\xi_i, \\ \text{s.t.} \quad & \boldsymbol{w}_-^T\boldsymbol{x}_i + b_- \geqslant 1 - \xi_i, \\ & \xi_i \geqslant 0, \quad i \in I^+, \end{aligned} \tag{9}$$

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

$$\begin{aligned} \max \quad & \sum_{j\in I^-}\alpha_j - \frac{1}{2}\sum_{j_1,j_2\in I^-}\alpha_{j_1}\alpha_{j_2}\boldsymbol{z}_{j_1}^T\left(\sum_{i\in I^+}\boldsymbol{z}_i\boldsymbol{z}_i^T\right)^{-1}\boldsymbol{z}_{j_2}, \\ \text{s.t.} \quad & 0 \leqslant \alpha_j \leqslant C_1, \quad j \in I^-, \end{aligned} \tag{10}$$

and

$$\begin{aligned} \max \quad & \sum_{i\in I^+}\alpha_i - \frac{1}{2}\sum_{i_1,i_2\in I^+}\alpha_{i_1}\alpha_{i_2}\boldsymbol{z}_{i_1}^T\left(\sum_{j\in I^-}\boldsymbol{z}_j\boldsymbol{z}_j^T\right)^{-1}\boldsymbol{z}_{i_2}, \\ \text{s.t.} \quad & 0 \leqslant \alpha_i \leqslant C_2, \quad i \in I^+, \end{aligned} \tag{11}$$

where $\boldsymbol{z}_k = (\boldsymbol{x}_k^T, 1)^T, k \in I$. After optimizing (10) and (11), we obtain the augmented vectors of the two nonparallel hyperplanes, which are:

[1] If $\sum_{i\in I^+}\boldsymbol{z}_i\boldsymbol{z}_i^T$ or $\sum_{j\in I^-}\boldsymbol{z}_j\boldsymbol{z}_j^T$ is ill-conditioned, we can add a regularization term $\epsilon E$, $\epsilon > 0$, where $E$ is an identity matrix of appropriate dimension. However, in the following, we shall continue to use $\sum_{i\in I^+}\boldsymbol{z}_i\boldsymbol{z}_i^T$ or $\sum_{j\in I^-}\boldsymbol{z}_j\boldsymbol{z}_j^T$ with the understanding that.

$$\boldsymbol{v}_+^T = (\boldsymbol{w}_+^T, b_+) = -\sum_{j\in I^-}\alpha_j \boldsymbol{z}_j^T \left(\sum_{i\in I^+}\boldsymbol{z}_i\boldsymbol{z}_i^T\right)^{-1}, \tag{12}$$

$$\boldsymbol{v}_-^T = (\boldsymbol{w}_-^T, b_-) = \sum_{i\in I^+}\alpha_i \boldsymbol{z}_i^T \left(\sum_{j\in I^-}\boldsymbol{z}_j\boldsymbol{z}_j^T\right)^{-1}. \tag{13}$$

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

$$\sum_{k\in I} w_{k,+}k(\boldsymbol{x}_k,\boldsymbol{x}) + b_+ = 0 \quad \text{and} \quad \sum_{k\in I} w_{k,-}k(\boldsymbol{x}_k,\boldsymbol{x}) + b_- = 0, \tag{14}$$

where $\boldsymbol{w}_\pm = (w_{1,\pm}, w_{2,\pm}, \ldots, w_{l,\pm})^T$ and $k(\boldsymbol{u},\boldsymbol{v})$ is some kernel. It is easy to obtain the similar primal QPPs for this nonlinear case; these QPPs are similar to (12) and (13). If we define $\boldsymbol{z}_k = (k(\boldsymbol{x}_1,\boldsymbol{x}_k),\ldots,k(\boldsymbol{x}_l,\boldsymbol{x}_k),1)^T$ for each $k \in I$, then we have similar dual QPPs to (10) and (11) and similar augmented vectors to (12) and (13) for (14).

The QPPs (10) and (11) have lower complexity than those of the classical SVM because they have only $l^\mp = |I^\mp|$ variables, whereas (4) has $l = l^+ + l^-$ variables, which allows TSVM to be approximately 4 times faster than classical SVM. This is because the complexity of classical SVM is no more than $\mathcal{O}(l^3)$, whereas the TSVM solves two QPPs, each of which is roughly of size $(l/2)$. Thus, the runtime ratio is approximately 4. It is necessary to point out that, to optimize (10) and (11), the linear TSVM requires two matrix inversions of size $(m+1)\times(m+1)$, whereas the nonlinear TSVM requires two matrix inversions of size $(l+1)\times(l+1)$.

The experiments have shown that the TSVM has good generalization performance and faster learning speed than the classical SVM. However, unlike the classical SVM, the TSVM only considers the empirical risk minimization principle in its objective functions. That is, in each QPP, it minimizes simultaneously the $L_2$-norm empirical risks of the examples in one class and the $L_1$-norm empirical risks of the examples in the other class with a trade-off parameter. This principle leads to two shortcomings when optimizing its objective functions. The first is that examples have small perpendicular distances from themselves to the two hyperplanes, which leads to relatively poor generalization performance. Another is that many training examples in the TSVM may become SVs, which may also reduce the generalization performance.

## 3. $\nu$-Twin support vector machine ($\nu$-TSVM)

As with the classical SVM, the trade-off factors $C_1$ and $C_2$ in the TSVM do not permit any other theoretical interpretation except for controlling the ratios of empirical risks. We introduce two new parameters $\nu_1$ and $\nu_2$ instead of the factors $C_1$ and $C_2$, and we rewrite the primal optimization problems (8) and (9) as follows:

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i\in I^+}\left(\boldsymbol{w}_+^T\boldsymbol{x}_i + b_+\right)^2 - \nu_1\rho_+ + \frac{1}{l^-}\sum_{j\in I^-}\xi_j, \\ \text{s.t.} \quad & -\boldsymbol{w}_+^T\boldsymbol{x}_j - b_+ \geqslant \rho_+ - \xi_j, \\ & \rho_+ \geqslant 0, \quad \xi_j \geqslant 0, \quad j\in I^-, \end{aligned} \tag{15}$$

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{j\in I^-}\left(\boldsymbol{w}_-^T\boldsymbol{x}_j + b_-\right)^2 - \nu_2\rho_- + \frac{1}{l^+}\sum_{i\in I^+}\xi_i, \\ \text{s.t.} \quad & \boldsymbol{w}_-^T\boldsymbol{x}_i + b_- \geqslant \rho_- - \xi_i, \\ & \rho_- \geqslant 0, \quad \xi_i \geqslant 0, \quad i\in I^+, \end{aligned} \tag{16}$$

There are two additional variables $\rho_\pm$ that need to be optimized in (15) and (16). To understand the roles of $\rho_\pm$, note that for all $\xi_j = 0, j \in I^-$ (or $\xi_i = 0, i \in I^+$), the negative (or positive) examples are separated by the positive (or negative) hyperplane, with the margin $\rho_+/(\boldsymbol{w}_+^T\boldsymbol{w}_+)$ (or $\rho_-/(\boldsymbol{w}_-^T\boldsymbol{w}_-)$). Also, in the constraints of (15) and (16), $\rho_\pm$ substitute for those of (8) and (9). The adaptive $\rho_\pm$ effectively overcomes the above shortcomings in the TSVM. Thus, one may expect that the $\nu$-TSVM obtains better generalization performance and fewer SVs than the TSVM.

Before studying the theoretical significance of $\nu_1$ and $\nu_2$, we first take the dual QPPs of (15) and (16) into account. The Lagrangian function of the problem (15) is as follows:

$$L(\boldsymbol{w}_+, b_+, \rho_+, \xi, \alpha, \boldsymbol{r}, s) = \frac{1}{2}\sum_{i\in I^+}\left(\boldsymbol{w}_+^T\boldsymbol{x}_i + b_+\right)^2 - \nu_1\rho_+ + \frac{1}{l^-}\sum_{j\in I^-}\xi_j - \sum_{j\in I^-}\alpha_j\left(-\boldsymbol{w}_+^T\boldsymbol{x}_j - b_+ - \rho_+ + \xi_j\right) - \sum_{j\in I^-} r_j\xi_j - s\rho_+, \tag{17}$$

where $s \geqslant 0, \alpha_j \geqslant 0, r_j \geqslant 0, j \in I^-$ are the Lagrangian multipliers. Differentiating the Lagrangian function (17) with respect to $\boldsymbol{w}_+$, $b_+$, $\rho_+$, and $\xi_j, j \in I^-$ yields the following KKT conditions:

$$\frac{\partial L}{\partial \boldsymbol{w}_+} = \sum_{i\in I^+}\left(\boldsymbol{w}_+^T\boldsymbol{x}_i + b_+\right)\boldsymbol{x}_i + \sum_{j\in I^-}\alpha_j\boldsymbol{x}_i = 0, \tag{18}$$

$$\frac{\partial L}{\partial b_+} = \sum_{i\in I^+}\left(\boldsymbol{w}_+^T\boldsymbol{x}_i + b_+\right) + \sum_{j\in I^-}\alpha_j = 0, \tag{19}$$

$$\frac{\partial L}{\partial \rho_+} = \sum_{j\in I^-} \alpha_j - v_1 - s = 0 \to \sum_{j\in I^-} \alpha_j \geqslant v_1, \tag{20}$$

$$\frac{\partial L}{\partial \xi_j} = \frac{1}{l^-} - \alpha_j - r_j = 0 \to 0 \leqslant \alpha_j \leqslant \frac{1}{l^-}, \quad j \in I^-. \tag{21}$$

Combining (18) and (19) leads to the same formulations (12) for computing the augmented vector of the positive hyperplane. Substituting the above KKT conditions into (17), we obtain the dual QPP of (15), which is

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{j_1,j_2\in I^-} \alpha_{j_1}\alpha_{j_2}\boldsymbol{z}_{j_1}^T\left(\sum_{i\in I^+}\boldsymbol{z}_i\boldsymbol{z}_i^T\right)^{-1}\boldsymbol{z}_{j_2}, \\ \text{s.t.} \quad & 0 \leqslant \alpha_j \leqslant \frac{1}{l^-}, \quad j \in I^-, \\ & \sum_{j\in I^-}\alpha_j \geqslant v_1. \end{aligned} \tag{22}$$

Similarly, the dual QPP of problem (16) is

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i_1,i_2\in I^+} \alpha_{i_1}\alpha_{i_2}\boldsymbol{z}_{i_1}^T\left(\sum_{j\in I^-}\boldsymbol{z}_j\boldsymbol{z}_j^T\right)^{-1}\boldsymbol{z}_{i_2}, \\ \text{s.t.} \quad & 0 \leqslant \alpha_i \leqslant \frac{1}{l^+}, \quad i \in I^+, \\ & \sum_{i\in I^+}\alpha_i \geqslant v_2. \end{aligned} \tag{23}$$

There are two differences between (22) and (10). First, there is an additional constraint $\sum_{j\in I^-}\alpha_j \geqslant v_1$ in (22). In fact, this constraint can be re-written as $\sum_{j\in I^-}\alpha_j = v_1$ because the optimal $\rho_+$ in (15) is actually larger than zero. Second, the linear term $\sum_{j\in I^-}\alpha_j$ no longer appears in the objective function of (22), i.e., it is quadratically homogeneous in $\alpha_j$. Similarly, we obtain the same results for the comparison in (23) and (11). To compute $\rho_\pm$, we choose the samples $\boldsymbol{x}_i$, $i \in I^+$ (or $\boldsymbol{x}_j$, $j \in I^-$) with $0 < \alpha_i < \frac{1}{l^+}$ (or $0 < \alpha_j < \frac{1}{l^-}$), which means $\xi_i = 0$ (or $\xi_j = 0$) and $\boldsymbol{w}_-^T\boldsymbol{x}_i + b_- = \rho_-$ (or $-\boldsymbol{w}_+^T\boldsymbol{x}_j - b_+ = \rho_+$) according to the KKT conditions. If we denote $n_\pm$ as the numbers of $\boldsymbol{x}_i$, $i \in I^\pm$ satisfied with $0 < \alpha_i < \frac{1}{l^\pm}$, then we have

$$\rho_+ = -\frac{1}{n_-}\sum_{j=1}^{n_-}(\boldsymbol{w}_+^T\boldsymbol{x}_j + b_+), \quad \rho_- = \frac{1}{n_+}\sum_{i=1}^{n_+}(\boldsymbol{w}_-^T\boldsymbol{x}_i + b_-). \tag{24}$$

We now turn to the theoretical interpretations for the parameters $v_1$ and $v_2$. To explain the theoretical significance of $v_1$ and $v_2$, let us first introduce the term *margin error* [29] as a sample $\boldsymbol{x}_k$ with $\xi_k > 0$; that is, this point is either an error or lies within the margin. Formally, the fractions of positive and negative margin errors are

$$\begin{aligned} R_{emp}^+ &= \frac{1}{l^+}\left|\left\{i | \boldsymbol{w}_-^T\boldsymbol{x}_i + b_- < \rho_-, i \in I^+\right\}\right|, \\ R_{emp}^- &= \frac{1}{l^-}\left|\left\{j | -\boldsymbol{w}_+^T\boldsymbol{x}_j - b_+ < \rho_+, j \in I^-\right\}\right|. \end{aligned} \tag{25}$$

Proposition 1 gives the theoretical interpretations of $v_1$ and $v_2$.

**Proposition 1.** *Suppose we run v-TSVM with l samples on dataset D, obtaining the result that $\rho_\pm > 0$. Then*

(i) *$v_2$ (or $v_1$) is an upper bound on the fraction of positive (or negative) margin errors.*
(ii) *$v_2$ (or $v_1$) is a lower bound on the fraction of positive (or negative) support vectors.*

**Proof.** The proof of this Proposition is similar to that of Proposition 5 in [29]. These results can be extended to the nonlinear case by considering the kernel function. □

## 4. Geometric interpretation of *v*-TSVM

In this section, we discuss the geometric interpretation of $v$-TSVM. Here, we focus only on (22) because both problems (22) and (23) have similar formulations.

Re-scaling $\alpha_j$ in (22) by $\alpha'_j = \alpha_j / v_1, j \in I^-$ leads to

$$\begin{aligned} \min \quad & \frac{v_1^2}{2}\sum_{j_1,j_2\in I^-} \alpha'_{j_1}\alpha'_{j_2}\boldsymbol{z}_{j_1}^T\left(\sum_{i\in I^+}\boldsymbol{z}_i\boldsymbol{z}_i^T\right)^{-1}\boldsymbol{z}_{j_2}, \\ \text{s.t.} \quad & 0 \leqslant \alpha'_j \leqslant \frac{1}{v_1 l^-}, \quad j \in I^-, \\ & \sum_{j\in I^-}\alpha'_j = 1. \end{aligned} \tag{26}$$

This is equivalent to the following re-scaling work in its primal problem (15)

$$\boldsymbol{w}'_+ = \frac{\boldsymbol{w}_+}{v_1}, \quad b'_+ = \frac{b_+}{v_1}, \quad \rho'_+ = \frac{\rho_+}{v_1}, \quad \xi'_j = \frac{\xi_j}{v_1}. \tag{27}$$

Note that if we use $\boldsymbol{w}_+/v_1$ and $b_+/v_1$ to substitute for $\boldsymbol{w}_+$, and $b_+$, the formulation of a hyperplane is in fact exactly equivalent. In the next equation, we still denote $\alpha'_j, \boldsymbol{w}'_+, b'_+, \rho'_+$, and $\xi'_j$ as $\alpha_j$, $\boldsymbol{w}_+$, $b_+$, $\rho_+$, and $\xi_j$ for the sake of brevity. Because $v_1 > 0$ is a constant, the objective function can be re-written to minimize

$$\frac{1}{2}\sum_{j_1 j_2 \in I^-} \alpha_{j_1}\alpha_{j_2}\boldsymbol{z}_{j_1}^T G^{-1}\boldsymbol{z}_{j_2}, \tag{28}$$

where the matrix $G = \sum_{i\in I^+}\boldsymbol{z}_i\boldsymbol{z}_i^T$ is symmetric and positive semi-definite.

Before discussing the geometric interpretation of $v$-TSVM, we first introduce the notion of *RCH* [6] for any point set $X$, which is,

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

$$RCH(X,\mu) = \left\{\boldsymbol{x} : \boldsymbol{x} = \sum_{i=1}^{n} a_i\boldsymbol{x}_i, \boldsymbol{x}_i \in X, \ \sum_{i=1}^{n} a_i = 1, \ 0 \leqslant a_i \leqslant \mu\right\}. \tag{29}$$

The properties of *RCH* can be found in Ref. [20].

Consider the following minimum generalized Mahalanobis-norm problem on $RCH(I^-,\mu_1)$

$$\begin{aligned} \min \quad & \frac{1}{2}\left\|\sum_{j\in I^-}\alpha_j\boldsymbol{z}_j\right\|_{GM}^2, \\ \text{s.t.} \quad & \sum_{j\in I^-}\alpha_j = 1, \quad 0 \leqslant \alpha_j \leqslant \mu_1, \quad j \in I^-, \end{aligned} \tag{30}$$

where the generalized Mahalanobis-norm is defined as $\|\boldsymbol{u}\|_{GM} = \sqrt{\boldsymbol{u}^T G^{-1}\boldsymbol{u}}$, and $\mu_1 = \frac{1}{v_1 l^-}$. Obviously, the objective function of this problem is consistent with that of (26). That is, (22) can be viewed as the minimum generalized Mahalanobis-norm problem on $RCH(I^-,\mu_1)$. In a similar manner, (23) can be viewed as the minimum generalized Mahalanobis-norm problem on $RCH(I^+,\mu_2)$ with $\mu_2 = \frac{1}{v_2 l^+}$, where this generalized Mahalanobis-norm is defined as $\|\boldsymbol{u}\|_{PM} = \sqrt{\boldsymbol{u}^T P^{-1}\boldsymbol{u}}$, $P = \sum_{j\in I^-}\boldsymbol{z}_j\boldsymbol{z}_j^T$.

## 5. Geometric algorithms for the TSVM

As has already been pointed out, an iterative geometric algorithm for solving the $\varepsilon$-optimal, $\varepsilon > 0$ as a low-threshold, minimum norm problem has been presented by Gilbert [11]:

**Algorithm 1: Gilbert's algorithm**

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

(2) *Stopping condition*: Find the point $\boldsymbol{x}_r$ such that $\boldsymbol{x}_r = \arg\min_{\boldsymbol{x}_i\in X}\{\langle\boldsymbol{x}_i,\boldsymbol{x}\rangle\}$, If the $\varepsilon$-optimality condition $\|\boldsymbol{x}\|_2^2 - \langle\boldsymbol{x}_r,\boldsymbol{x}\rangle < \varepsilon$ holds, then the point $\boldsymbol{x}$ is the $\varepsilon$-optimal solution. Otherwise, go to Step (3).

(3) *Adaptation*: Set $q = \frac{\langle\boldsymbol{x},\boldsymbol{x}-\boldsymbol{x}_r\rangle}{\|\boldsymbol{x}-\boldsymbol{x}_r\|_2^2}$ if $\boldsymbol{x} - \boldsymbol{x}_r \neq \boldsymbol{0}$, otherwise $q = 0$; set $\boldsymbol{x} = \boldsymbol{x} + q(\boldsymbol{x}_r - \boldsymbol{x})$. Continue with Step (2).

To optimize the minimum generalized Mahalanobis-norm problems on $RCH(I^-,\mu_1)$ and $RCH(I^+,\mu_2)$, an important task is to find the representations of extreme points in $RCH(I^-,\mu_1)$ and $RCH(I^+,\mu_2)$. Mavroforakis and Theodoridis [20] have pointed out that the extreme point of $RCH(X,\mu)$ is the combination of some points in $X$:

**Proposition 2.** *The extreme point of the RCH(X, μ) has coefficient $a_i$ belonging to the set $S = \{0, \mu, 1 - \lfloor 1/\mu\rfloor\mu\}$, i.e., each extreme point of RCH(X, μ) is the convex combination of $m = \lceil 1/\mu\rceil$ distinct 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 $a_m = 1 - \lfloor 1/\mu\rfloor\mu$.*

Proposition 2 provides a necessary but not sufficient condition for determining the extreme points of an *RCH*, in which the number of points satisfying the condition is much larger than of the number of extreme points. Therefore, it is impossible to inspect all candidate points in the optimization process. Fortunately, it does not need to consider all possible candidate points in the geometric algorithm, but it instead only needs to compute the generalized Mahalanobis-norm projections of

all negative (or positive) samples, and choose the first $\lceil 1/\mu_1 \rceil$ (or $\lceil 1/\mu_2 \rceil$) points with the smallest generalized Mahalanobis-norm projections to update the current point. Based on the above discussions, we obtain the geometric algorithm for the TSVM (GA-TSVM) as follows (only for finding the positive hyperplane).

**Algorithm 2: Geometric algorithm for the TSVM (GA-TSVM)**

(1) *Initialization*: Set parameters $1/|I^-| \leqslant \mu_1 \leqslant 1$, $\lambda_1 = 1 - \lfloor 1/\mu_1 \rfloor$, and $m_1 = \lceil 1/\mu_1 \rceil$; set the initial point $\boldsymbol{z}$ as the centroid point of the corresponding *RCH*, i.e., set $\alpha_j = 1/|I^-|$, $j \in I^-$ for all negative examples.

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

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

This linear GA-TSVM can be easily extended to the nonlinear case by introducing a kernel function. This case has almost the same complexity as Gilbert's (the extra cost is the sort involved in each step to find the least $\lceil 1/\mu_1 \rceil$ (or $\lceil 1/\mu_2 \rceil$) inner products, plus the cost to evaluate the inverse of $G$); the same caching scheme can be used, with only $\mathcal{O}(l^{\pm})$ storage requirements (if the inverse of $G$ is stored, the storage requirements will increase to $\mathcal{O}(l^2)$ for the nonlinear case.).

As with the geometric algorithms for the classical SVM, this GA-TSVM is efficient for learning the nonparallel hyperplanes. However, in Step 2 of Algorithm 2, all negative (or positive) examples need to be inspected to find the best $\boldsymbol{z}_r^{opt}$, which is time-consuming for large-scale or high-dimensional practical problems. Here, we introduce a *probabilistic speed-up strategy* to degrade the kernel computation complexity.

Generally, in the GA-TSVM, the impacts of examples that are far from the minimum norm point become smaller and smaller during the learning process. Specifically, if an example has no impact on the minimum norm point, such as the non-support vector (NSV), its coefficient will decrease iteratively to zero. Therefore, we consider the probabilistic speed-up strategy, which neglects as many NSVs as possible in the inspection work according to the coefficients: For the negative (or positive) samples, randomly choose some of the examples with "probability" $\alpha_j, j \in I^-$ (or $\alpha_i, i \in I^+$) in the iterative process, such that the sum of coefficients of these examples is at least $100\cdot(1-\zeta)\%$, $0 < \zeta < 1$, and select the first $\lceil 1/\mu_1 \rceil$ (or $\lceil 1/\mu_2 \rceil$) examples with the smallest generalized Mahalanobis-norm projections. This probabilistic speed-up strategy is very effective for reducing kernel evaluations, especially at the end stage of learning. This is because the coefficients of most NSVs are getting smaller and smaller and close to zero during the end stage of the learning process, which means that we only need to inspect SVs in the end stage of the learning process. To summarize, our probabilistic speed-up strategy procedure is as follows:

**Procedure 1** (*Probabilistic speed-up strategy*).

(1) Set $\widehat{I}^{\pm} = \phi$ as the index set of examples being selected in the positive/negative samples. Set $s_0^{\pm} = \sum_{i\in I^{\pm}} \alpha_i, \alpha'_i = \alpha_i, i \in I^{\pm}, \hat{s}^{\pm} = 0$, and $0 < \zeta < 1$;

(2) **While** $\hat{s}^{\pm} < 1 - \zeta$ or $|\widehat{I}^{\pm}| < \lceil 1/\mu_{1,2} \rceil$ **Do**

(2.1) Generate a uniformly distributed random number $r$ in the interval (0,1), and find the index $j \in I^{\pm}$ of the next selected example as the smallest integer in $I$ such that

$$\sum_{j'=1}^{j} \alpha'_{j'} > r; \tag{31}$$

(2.2) Set $s_0^{\pm} = s_0^{\pm} - \alpha_j, \hat{s}^{\pm} = \hat{s}^{\pm} + \alpha_j, I^{\pm} = I^{\pm} - \{j\}$, and $\widehat{I}^{\pm} = \widehat{I}^{\pm} + \{j\}$, and normalize $\alpha' i$, $i \in I^{\pm}$, as $\alpha\prime_i = \alpha_i / s_0^{\pm}$;

(3) Find $\boldsymbol{z}_r^{opt}$, composed of the $\lceil 1/\mu_{1,2} \rceil$ samples with smallest projections whose indices are in $\widehat{I}^{\pm}$.

Combining the GA-TSVM with this procedure leads to a probabilistic speed-up GA-TSVM (PGA-TSVM). The experiments in the next section show that the PGA-TSVM effectively reduces the number of kernel evaluations without loss of accuracy. Note that this probabilistic speed-up procedure is still time-consuming for determining the index sets $\widehat{I}^{\pm}$ if we directly run it. This is because very small coefficients, such as those of the NSVs, will increase the search depth of this procedure. To account for this problem, we employ a simple *Sorting Method* that maintains the sample order in approximate sort order according to the coefficients. In this method, we move up the positions of the $\lceil 1/\mu_1 \rceil$ (or $\lceil 1/\mu_2 \rceil$) examples if these examples are selected to construct $\boldsymbol{z}_r^{opt}$ in the current iteration. This moves examples with larger coefficients toward the top of the sample search list, effectively reducing the search depth of iterations. We point out that this sorting method does not sort the examples strictly according to their coefficients in the iterative process, but it can approximately sort all examples in $I^{\pm}$ in descending order according to their multipliers. Because the sorting only requires a swap of memory addresses, it adds negligible overhead to the learning, as we will show in our performance analysis.

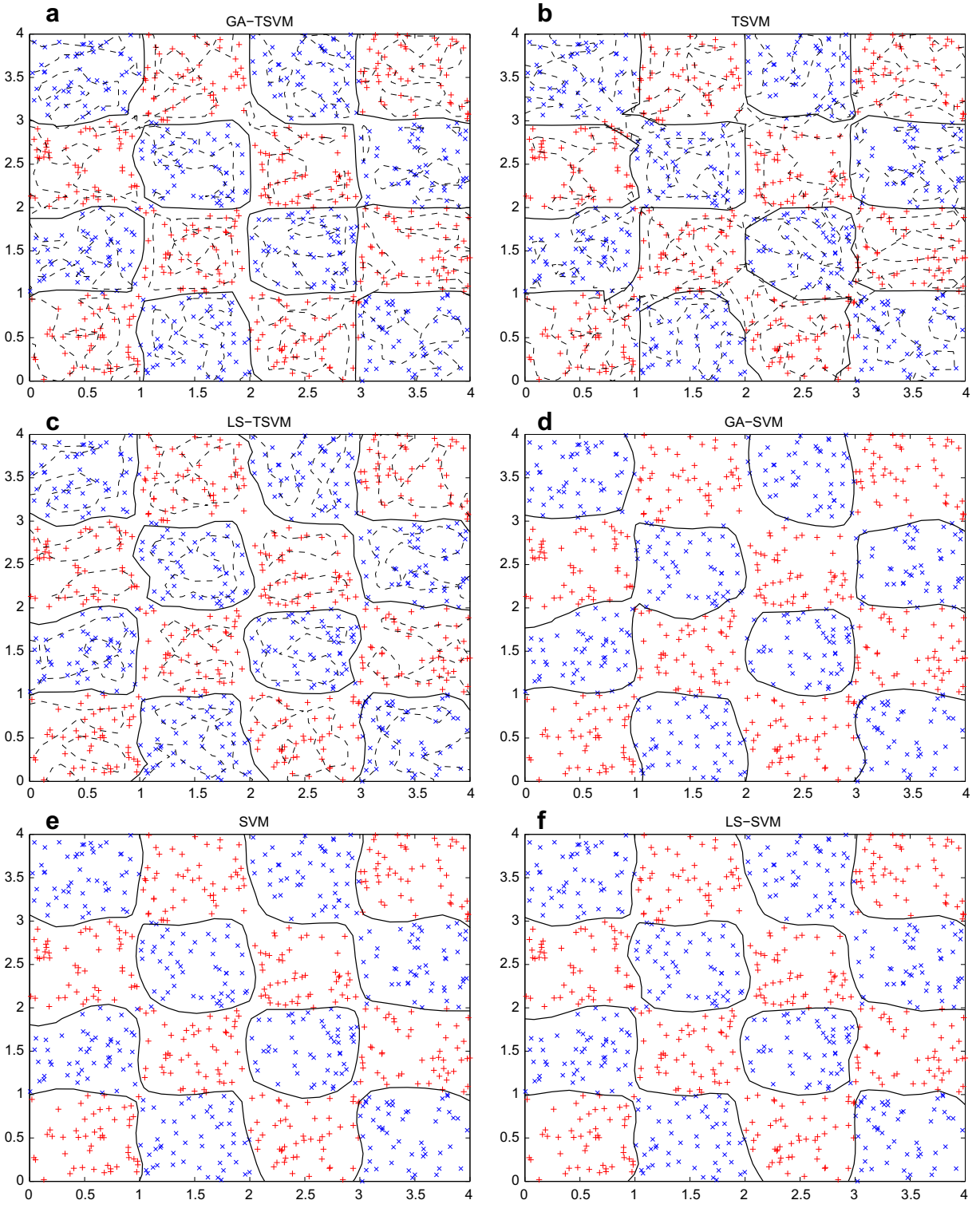


**Fig. 1.** Classification results of the GA-TSVM, TSVM, LS-TSVM, GA-SVM, SVM, and LS-SVM on the checkerboard dataset. In The GA-TSVM, TSVM, and LS-TSVM, the pairs of nonparallel hyperplanes and separating hyperplanes are shown as dashed and solid curves, respectively. In the GA-SVM, SVM, and LS-SVM, the separating hyperplanes are shown as solid curves.

**Table 1**
Performance on checkerboard dataset.

| Algorithm | Accuracy (%) | Kernel evaluations | Number of SVs | CPU time (sec) |
|---|---|---|---|---|
| GA-TSVM | 95.37 | 3.1201e+08 | 80 + 75[a] | 9.6410 |
| PGA-TSVM | 95.28 | 1.8592e+08 | 78 + 77 | 6.1520 |
| GA-SVM | 93.66 | 6.8493e+09 | 143 | 210.0470 |
| TSVM | 93.13 | – | 360 + 356 | 13.4220 |
| SVM | 92.91 | – | 141 | – |
| LS-TSVM | 94.81 | – | 400 + 400 | 0.7250 |
| LS-SVM | 94.76 | – | 800 | 0.5780 |

[a] $n_1+n_2$: $n_1$ and $n_2$ are the numbers of SVs of the two QPPs.

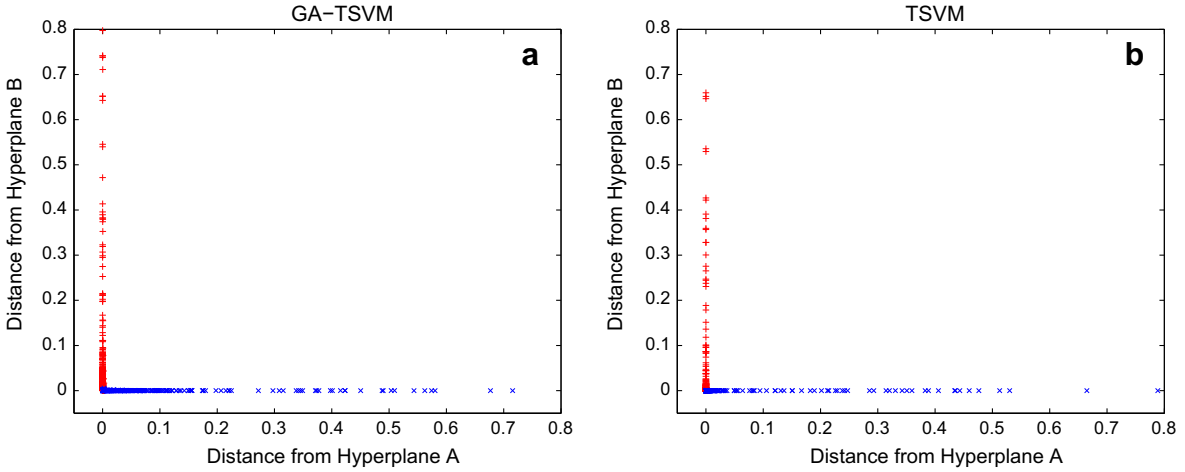


**Fig. 2.** Two-dimensional projections for test points from the checkerboard dataset, using the GA-TSVM and TSVM.

## 6. Experiments and discuss

In this section, we compare the performance of the GA-TSVM and PGA-TSVM to that of the GA-SVM [20], LS-SVM, LS-TSVM, TSVM, and SVM. All these algorithms are implemented in Matlab 6.5 [2] except for The SVM, which is implemented by LIBSVM.[3] We test their performance on several artificial and benchmark datasets,[4] which are commonly used for testing machine-learning algorithms. For all geometric algorithms, we set $\varepsilon$ to 0.001 and fix the parameter $\zeta$ in the PGA-TSVM to 0.05. For the LS-SVM and LS-TSVM, we use the Matlab function "*inv.m*" to obtain the solutions, whereas for the TSVM, we use the Matlab function "*quadprog.m*" as the optimization tool. Note that in these geometric algorithms, the total computational costs in the experiments are directly affected by the numbers of kernel evaluations. Based on the above discussions, we demonstrate the performance of these geometric algorithms in terms of test accuracy, kernel evaluations, number of SVs, and CPU time for learning. To fairly reflect the performance of these algorithms, in the simulations, the parameters $C$ and $\gamma$ are selected from the set of values $\{2^i | i = -9, -8, \ldots, 10\}$, whereas $\mu$ are selected from the set of values $\{2^i | i = -10, -9, \ldots, 0\}$, by tuning a set comprising a random 10% of the examples. Also, we set $\mu_1 = \mu_2$ in the GA-TSVM and PGA-TSVM, and $C_1 = C_2$ in the LS-TSVM and TSVM in order to reduce the complexity of parameter selection.

### 6.1. Artificial dataset

To illustrate graphically the effectiveness of the GA-TSVM and PGA-TSVM, we test their ability to learn the artificial checkerboard dataset. The checkerboard dataset consists of a series of uniform points in $\mathcal{R}^2$ with red and blue points taken from the 16 red and blue squares of a checkerboard. This is a tricky test case in the data mining field for testing the performance of nonlinear classifiers.

Fig. 1 shows the simulation results of the GA-TSVM, TSVM, LS-TSVM, GA-SVM, SVM, and LS-SVM with 800 uniform training points, each square consisting of 50 points. Note that the classification result of the PGA-TSVM is not given because it is

[2] Available at: http://www.mathworks.com.
[3] Available at: http://www.csie.ntu.edu.tw/cjlin/libsvm/.
[4] Available at: http://www.ics.uci.edu/mlearn/MLRepository.html, and http://ida.first.fraunhofer.de/projects/bench/.

**Table 2**
Performance on benchmark datasets.

| Dataset | | GA-TSVM | PGA-TSVM | GA-SVM | TSVM | SVM | LS-TSVM | LS-SVM |
|---|---|---|---|---|---|---|---|---|
| Heart (270 × 14) | C1 | 84.08 ± 4.96 | 83.14 ± 5.27 | 81.85 ± 6.81 | 82.63 ± 5.59 | 81.76 ± 6.12 | 83.81 ± 4.81 | 82.59 ± 5.64 |
| | C2 | 2.74e+6 | 1.98e+6 | 8.92e+7 | – | – | – | – |
| | C3 | 70.5 + 108.3 | 72.6 + 110.8 | 157.1 | 94.3 + 113.7 | 155.6 | – | – |
| | C4 | 0.78 | 0.57 | 3.69 | 1.62 | – | 0.01 | 0.02 |
| German (1000 × 20) | C1 | 76.25 ± 3.67 | 75.94 ± 3.84 | 72.57 ± 3.40 | 75.51 ± 3.41 | 72.32 ± 3.58 | 74.46 ± 3.19 | 72.75 ± 3.83 |
| | C2 | 2.06e+7 | 1.15e+7 | 7.14e+9 | – | – | – | – |
| | C3 | 93.5 + 153.6 | 94.6 + 151.7 | 535.4 | 248.5 + 536.2 | 541.8 | – | – |
| | C4 | 13.89 | 7.76 | 51.41 | 22.66 | – | 0.54 | 0.77 |
| Twonorm (400 × 20) | C1 | 97.25 ± 2.02 | 96.75 ± 2.46 | 93.50 ± 3.16 | 96.89 ± 2.19 | 93.68 ± 2.78 | 94.62 ± 3.87 | 95.25 ± 2.75 |
| | C2 | 3.08e+6 | 1.76e+6 | 6.73e+7 | – | – | – | – |
| | C3 | 42.6 + 36.8 | 44.1 + 37.5 | 350.2 | 185.7 + 173.4 | 341.5 | – | – |
| | C4 | 1.35 | 0.79 | 13.32 | 4.29 | – | – | – |
| Diabetes (768 × 8) | C1 | 76.37 ± 2.85 | 76.21 ± 3.67 | 71.98 ± 3.66 | 75.97 ± 3.08 | 72.42 ± 3.13 | 74.86 ± 3.09 | 73.64 ± 2.87 |
| | C2 | 4.62e+6 | 2.98e+6 | 7.96e+8 | – | – | – | – |
| | C3 | 107.3 + 112.4 | 109.1 + 110.9 | 443.7 | 228.5 + 394.1 | 440.2 | – | – |
| | C4 | 2.18 | 1.51 | 41.52 | 12.1433 | – | 0.29 | 0.37 |
| Thyroid (215 × 5) | C1 | 98.18 ± 3.18 | 97.56 ± 3.59 | 95.82 ± 4.26 | 97.45 ± 4.79 | 96.09 ± 4.02 | 96.27 ± 3.49 | 98.11 ± 3.43 |
| | C2 | 1.47e+7 | 8.71e+6 | 2.08e+8 | – | – | – | – |
| | C3 | 56.0 + 33.2 | 57.8 + 38.9 | 171.3 | 58.9 + 132.5 | 175.1 | – | – |
| | C4 | 0.44 | 0.36 | 2.51 | 0.96 | – | 0.02 | 0.02 |
| WDBC (569 × 30) | C1 | 98.25 ± 1.43 | 96.32 ± 2.98 | 95.09 ± 1.99 | 96.67 ± 2.10 | 95.71 ± 2.13 | 96.49 ± 2.19 | 97.90 ± 1.71 |
| | C2 | 5.95e+6 | 3.16e+6 | 9.98e+7 | – | – | – | – |
| | C3 | 47.9 + 21.4 | 45.7 + 23.5 | 83.4 | 315.2 + 187.0 | 78.7 | – | – |
| | C4 | 3.59 | 2.35 | 28.74 | 9.51 | – | 0.15 | 0.37 |
| Ionosphere (351 × 34) | C1 | 94.72 ± 2.76 | 93.31 ± 3.64 | 89.17 ± 3.77 | 94.44 ± 2.85 | 90. 26 ± 3.17 | 92.40 ± 3.66 | 92.13 ± 1.76 |
| | C2 | 3.70e+7 | 2.59e+7 | 6.61e+8 | – | – | – | – |
| | C3 | 64.5 + 109.7 | 66.9 + 112.4 | 278.2 | 175.3 + 104.7 | 267.5 | – | – |
| | C4 | 3.24 | 2.76 | 46.41 | 14.47 | – | 0.03 | 0.04 |
| Pima Indian (768 × 8) | C1 | 74.68 ± 2.76 | 75.13 ± 2.89 | 76.38 ± 4.23 | 76.34 ± 3.51 | 76. 71 ± 3.92 | 75.53 ± 3.47 | 77.24 ± 4.54 |
| | C2 | 1.45e+8 | 9.76e+7 | 3.85e+9 | – | – | – | – |
| | C3 | 153.6 + 209.1 | 151.5 + 201.8 | 430.9 | 164.3 + 304.5 | 416.4 | – | – |
| | C4 | 7.67 | 6.14 | 68.31 | 24.73 | – | 0.31 | 0.38 |
| Sonar (208 × 60) | C1 | 91.71 ± 5.52 | 91.02 ± 6.31 | 91.43 ± 5.86 | 90.62 ± 6.81 | 91. 75 ± 7.12 | 90.05 ± 5.24 | 91.16 ± 7.12 |
| | C2 | 2.60e+7 | 1.59e+7 | 1.20e+8 | – | – | – | – |
| | C3 | 49.9 + 48.6 | 52.7 + 50.1 | 168.6 | 94.4 + 80.6 | 162.3 | – | – |
| | C4 | 0.52 | 0.39 | 2.34 | 0.74 | – | 0.01 | 0.01 |

C1, C2, C3, and C4 represent the test accuracy (%), number kernel evaluations, number of SVs, and CPU time (sec), respectively.

similar to that of the GA-TSVM. It can be seen that the GA-TSVM obtains the best separating hyperplane among these six algorithms. This indicates that our algorithms effectively improve the performance of the TSVM. To reflect the performance of these algorithms, we test the classification accuracies on 6400 uniform test examples. Table 1 lists the prediction accuracy, learning CPU time, number of SVs, and number of kernel evaluations of these algorithms. It can be seen that the proposed GA-TSVM and PGA-TSVM obtain better classification performance than the other classifiers. Compared with the kernel evaluations of the three geometric algorithms, it can also be noted that both the GA-TSVM and PGA-TSVM require fewer kernel evaluations than the GA-SVM. In particular, the PGA-TSVM only needs about 2.71% as many kernel evaluations as the GA-SVM. We do not count the kernel evaluations of the other four algorithms because they are optimized by using Matlab functions or LIBSVM. In addition, we find that our GA-TSVM and PGA-TSVM have fewer SVs than the TSVM, indicating that the introduced parameters $\nu$'s and $\rho$'s can effectively reduce the number of SVs. As for the learning CPU time, it can be seen that our GA-TSVM and PGA-TSVM take less learning time than the GA-SVM. Furthermore, we find that the PGA-TSVM uses about 63.81% as much CPU time as the GA-TSVM, and the PGA-TSVM performs about 59.39% as many kernel evaluations as the GA-TSVM. This is because the PGA-TSVM needs to generate random numbers and sort all examples according to their coefficients in the iterative process.

To further illustrate the performance of the GA-TSVM and TSVM, Fig. 2 shows two-dimensional scatter plots of 800 test data points (class +: 400, class −: 400) for this checkerboard dataset for the GA-TSVM and TSVM. The plots are obtained by plotting points with coordinates $(d^i_+, d^i_-)$, where $d^i_+$ and $d^i_-$ are the respective distances of a test point $\boldsymbol{x}_i$ to the positive and negative hyperplanes. In the figures, the point $\boldsymbol{x}_i$ is assigned to Class + if the value of $d^i_+$ is less than $d^i_-$, and vice versa. In Fig. 2, each sample is plotted as a "+" if its class label is +1, whereas it is plotted as a "×" if its class label is −1. Hence, the clusters of points indicate how well the classification criterion is able to discriminate between the two classes. From Fig. 2, it can be seen that, for both methods, the largest number of test examples are clustered around the corresponding

positive and negative hyperplanes. However, for the TSVM, the distances from the two hyperplanes to most test examples are very small (near zero), whereas for the GA-TSVM, most test examples are at a large distances to the other hyperplanes. This indicates that the GA-TSVM obtains better generalization than the TSVM.

### 6.2. Benchmark datasets

To further test the performance of these algorithms, we run them on several publicly available benchmark datasets, including the Heart, German, Twonorm, Diabetes, Thyroid, Ionosphere, Wisconsin diagnostic breast cancer (WDBC), Pima Indian, and Sonar datasets. These datasets are commonly used in testing machine-learning algorithms. Note that all samples are normalized such that the continuous features are located in the range [0, 1] before learning. In the simulations, we only consider the Gaussian kernel for these classifiers.

Table 2 presents the average learning results of these seven algorithms with 10-fold cross-validation on these benchmark datasets. It can be seen that our GA-TSVM and PGA-TSVM demonstrate better test accuracy than the other algorithms. In addition, both algorithms, especially the proposed PGA-TSVM, require fewer kernel evaluations than the GA-SVM on these datasets. It can also be seen that our GA-TSVM and PGA-TSVM take less learning CPU time than the GA-SVM, whereas the TSVM spends more time learning than the GA-SVM. This is because we only use the Matlab function to train the TSVM. Table 2 also compares the numbers of SVs of the first five algorithms. It can be seen that the GA-TSVM and PGA-TSVM have many fewer SVs than the TSVM. This is because the constraints in the TSVM – that the distances from a hyperplane to the samples of the other class are at least one – are over-restrictive for most samples, which causes most samples to be located in the

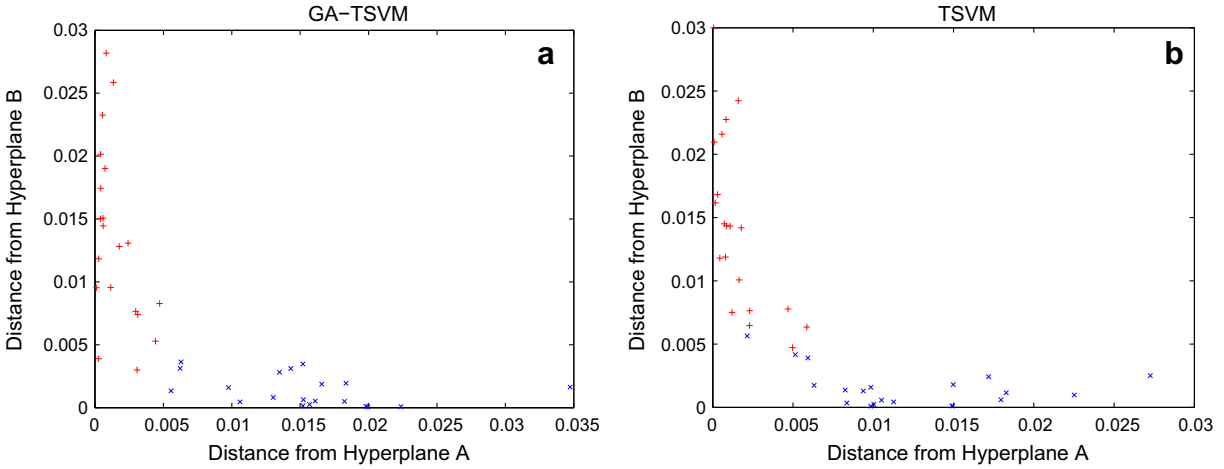


**Fig. 3.** Two-dimensional projections for test points from the Twonorm dataset, using the GA-TSVM and TSVM.

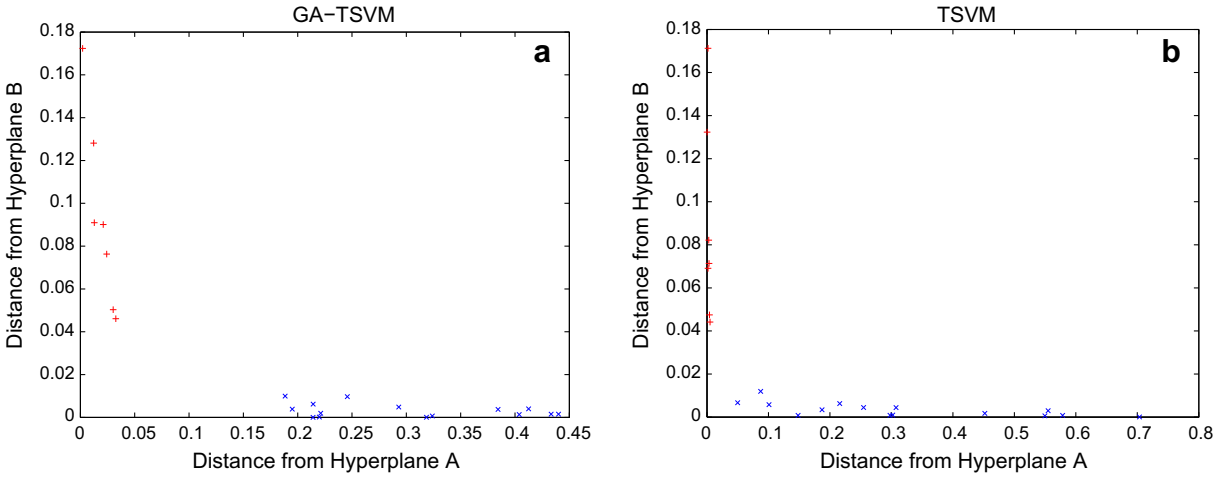


**Fig. 4.** Two-dimensional projections for test points from the Thyroid dataset, using the GA-TSVM and TSVM.

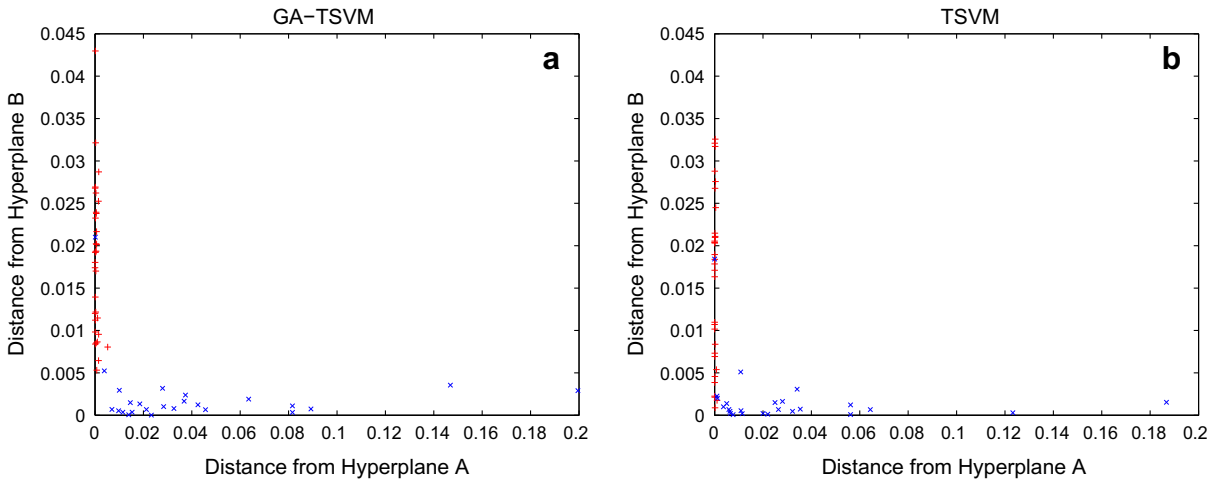


**Fig. 5.** Two-dimensional projections for test points from the WDBC dataset, using the GA-TSVM and TSVM.

margins. We should point out that a comparison of the number of SVs between TSVMs and SVMs would be meaningless because TSVMs lose sparsity compared with SVMs.

Figs. 3–5 show the two-dimensional scatter plots of the test data points (comprising 10% of the data points) for the Two-norm, Thyroid, and WDBC datasets, using the GA-TSVM and TSVM classifiers. The plots are obtained by plotting points with coordinates $(d_+^i, d_-^i)$. From these Figures, it can be seen that our GA-TSVM obtains better separation than the TSVM because the projections of the two classes obtained by the GA-TSVM are more distinct than those of the TSVM.

## 7. Conclusions

In this paper, we presented several extensions to the TSVM. First, we introduced a novel $v$-TSVM for the TSVM. Similar to the $v$-SVM, the parameters $v$s in the $v$-TSVM have greater theoretical significance, such as the determination of the bounds for the fractions of SVs and margin errors. Furthermore, the $v$-TSVM effectively overcomes a shortcoming of the TSVM, in which the constraints are often over-restricted, by introducing a pair of adaptive $\rho_{\pm}$. Therefore, it can effectively reduce the number of SVs.

We also discussed the geometric interpretation of the $v$-TSVM so that it can be interpreted as two minimum generalized Mahalanobis-norm problems on two RCHs. Meanwhile, we introduced the geometric algorithm for optimizing the TSVM according to the well-known Gilbert's algorithm. To improve the efficiency of the GA-TSVM, we presented a sped-up PGA-TSVM by integrating the probabilistic speed-up strategy and sorting method. The experimental results on several datasets showed that the GA-TSVM and PGA-TSVM obtain better generalization performance and have faster learning speed than the classical SVM and TSVM classifiers.

## Acknowledgements

This work has been partly supported by the Shanghai Leading Academic Discipline Project (S30405) and the Natural Science Foundation of Shanghai Normal University (SK200937).

## References

[1] K.P. Bennett, E.J. Bredensteiner, Duality and geometry in SVM classifiers, in: P. Langley (Ed.), Proceedings of the 17th International Conference on Machine Learning, Morgan Kaufmann, Los Altos, CA, 2000, pp. 57–64.
[2] M.P.S. Brown, W.N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T.S. Furey, J.M. Ares, D. Haussler, Knowledge-based analysis of microarray gene expression data by using support vector machine, Proceedings of the National Academy of Sciences of the United States of America 97 (1) (2000) 262–267.
[3] X.B. Cao, Y.W. Xu, D. Chen, H. Qiao, Associated evolution of a support vector machine-based classifier for pedestrian detection, Information Sciences 179 (8) (2009) 1070–1077.
[4] V. Christianini, J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, Cambridge, 2002.
[5] C. Cortes, V.N. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.
[6] D.J. Crisp, C.J.C. Burges, A geometric interpretation of $v$-SVM classifiers, in: S. Solla, T. Leen, K.-R. Muller (Eds.), Advances in Neural Information Processing Systems, vol. 12, 2000, pp. 244–250.
[7] I. El-Naqa, Y. Yang, M.N. Wernik, N.P. Galatsanos, R.M. Nishikawa, A support vector machine approach for detection of microcalcifications, IEEE Transactions on Medical Imaging 21 (12) (2002) 1552–1563.
[8] V. Franc, V. Hlaváč, An iterative algorithm learning the maximal margin classifier, Pattern Recognition 36 (9) (2003) 1985–1996.
[9] S. Ghorai, S.J. Hossain, P.K. Dutta, A. Mukherjee, Newton's method for nonparallel plane proximal classifier with unity norm hyperplanes, Signal Processing 90 (1) (2010) 93–104.
[10] S. Ghorai, A. Mukherjee, P.K. Dutta, Nonparallel plane proximal classifier, Signal Processing 89 (4) (2009) 510–522.

[11] E.G. Gilbert, An iterative procedure for computing the minimum of a quadratic form on a convex set, SIAM Journal on Control 4 (1) (1966) 61–79.
[12] Jayadeva, R. Khemchandani, S. Chandra, Twin support vector machines for pattern classification, IEEE Transactions on Pattern Analysis and Machine Intellegence 29 (5) (2007) 905–910.
[13] J.-T. Jeng, C.-C. Chuang, S.-F. Su, Support vector interval regression networks for interval regression analysis, Fuzzy Sets and Systems 138 (2) (2003) 283–300.
[14] T. Joachims, C. Ndellec, C. Rouveriol, Text categorization with support vector machines: learning with many relevant features, in: European Conference on Machine Learning, vol. 10, Chemnitz, Germany, 1998, pp. 137–142.
[15] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy, A fast iterative nearest point algorithm for support vector machine classifier design, IEEE Transactions on Neural Networks 11 (1) (2000) 124–136.
[16] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy, Improvements to platt's SMO algorithm for SVM classifier design, Neural Computation 13 (3) (2001) 637–649.
[17] M.A. Kumar, M. Gopal, Application of smoothing technique on twin support vector machines, Pattern Recognition Letter 29 (13) (2008) 1842–1848.
[18] M.A. Kumar, M. Gopal, Least squares twin support vector machines for pattern classification, Expert Systems with Applications 36 (4) (2009) 7535–7543.
[19] O.L. Mangasarian, E.W. Wild, Multisurface proximal support vector classification via generalized eigenvalues, IEEE Transactions on Pattern Analysis and Machine Intellegence 28 (1) (2006) 69–74.
[20] M.E. Mavroforakis, S. Theodoridis, A geometric approach to support vector machine (SVM) classification, IEEE Transactions on Neural Networks 17 (3) (2007) 671–682.
[21] J. Mercer, Functions of positive and negative type and the connection with the theory of integal equations, Philosophical Transactions of the Royal Society of London, Series A 209 (1909) 415–446.
[22] S. Mukherjee, E. Osuna, F. Girosi, Nonliner prediction of chaotic time series using a support vector machine, in: Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing, Amelia Island, FL, USA, 1997, pp. 511–520.
[23] E. Osuna, R. Freund, F. Girosi, Training support vector machines: An application to face detection, in: Proceedings of IEEE Computer Vision and Pattern Recognition, San Juan, Puerto Rico, 1997, pp. 130–136.
[24] E. Osuna, R. Freund, F. Girosi, Support vector machines: training and applications, Technical Report, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1997.
[25] X. Peng, TSVR: An efficient twin support vector machine for regression, Neural Networks 23 (3) (2010) 365–372.
[26] J. Platt, Fast training of support vector machines using sequential minimal optimization, in: B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods-Support Vector Learning, MIT Press, Cambridge, MA, 1999, pp. 185–208.
[27] J. Ramírez, J.M. Górriz, D. Salas-Gonzalez, A. Romero, M. López, I. Álvarez, M. Gómez-Río, Computer-aided diagnosis of Alzheimer's type dementia combining support vector machines and discriminant set of features, Information Sciences (2009), doi:10.1016/j.ins.2009.05.012.
[28] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, 1996.
[29] B. Schölkopf, A. Smola, R. Williamson, P.L. Bartlett, New support vector algorithms, Neural Computation (2000) 1083–1121.
[30] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process Letter 9 (3) (1999) 293–300.
[31] J.A.K. Suykens, L. Lukas, P. van Dooren, B. De Moor, J. Vandewalle, Least squares support vector machine classifiers: A large scale algorithm, in: Proceedings of European Conference of Circuit Theory Design, 1999, pp. 839–842.
[32] Q. Tao, G. Wu, J. Wang, A generalized S–K algorithm for learning $v$-SVM classifiers, Pattern Recognition Letter 25 (10) (2004) 1165–1171.
[33] Q. Tao, G. Wu, J. Wang, A general soft method for learning SVM classifiers with $L_1$-norm penalty, Pattern Recognition 41 (3) (2008) 939–948.
[34] V.N. Vapnik, The Natural of Statistical Learning Theory, Springer, New York, 1995.
[35] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[36] J. Wang, Q. Tao, J. Wang, Kernel projection algorithm for large-scale SVM problems, Journal of Computer Science and Technology 17 (5) (2002) 556–564.
[37] D. Zhou, B. Xiao, H. Zhou, et al., Global geometric of SVM classifiers, Technical Report, AI Lab, Institute of automation, Chinese Academy of Sciences, 2002.