Applied Soft Computing Journal 101 (2021) 107071

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Condensing the solution of support vector machines via radius-margin bound

Xiaoming Wang [a,*], Shitong Wang [b], Zengxi Huang [a], Yajun Du [a]

[a] School of Computer and Software Engineer, Xihua University, Chengdu 610039, China
[b] School of Digital Media, Jiangnan University, Wuxi 214122, China

ARTICLE INFO

Article history:
Received 5 May 2020
Received in revised form 23 November 2020
Accepted 25 December 2020
Available online 30 December 2020

Keywords:
Support vector machines
Reduced support vector machines
Sparse solution
Radius-margin bound

ABSTRACT

The paper deals with the problem of how to condense efficiently the used basis vectors in the decision function of support vector machines (SVM). Most existing methods learn the basis vectors and the corresponding coefficients by maximizing the margin between different classes. They ignore the key fact that condensing the solution of SVM is equivalent to constructing the SVM model in the transformation space determined by the basis vectors. Thus, the radius-margin bound, which is related with the generalization ability of SVM, changes with them. In the paper, we propose a novel method called sparse support vector machine guided by radius-margin bound (RMB-SSVM) to learn the condensed solution for SVM. The key characteristic of RMB-SSVM is that it employs a criterion that exploits integratedly the margin and the radius of the minimum hypersphere enclosing the data to guide the selecting of the basis vectors and the learning of the corresponding coefficients. Thus, the learning criterion of RMB-SSVM is directly related with the generalization ability of SVM and so it can yield better performance. We develop an effective method to handle the proposed optimization model, and propose a heuristic scheme to select the basis vectors used in the decision function. Further, we explore how to use the maximum pairwise distance over the pairs of data points to approximate the radius in our methodology. This can simplify the proposed model and reduce the training time while not drastically impairing the performance. Finally, we conduct the comprehensive experiments to demonstrate the effectiveness and superiority of the proposed methods by comparing them with the counterparts.

1. Introduction

Support vector machines (SVM) [1,2] has attracted a lot of attention in the pattern recognition community due to its good generalization performances [3–9]. In order to handle the binary classification task, SVM constructs a decision hyperplane (function) that separates the two classes of data points. The decision function is determined only by the so-called support vectors (SVs) and their corresponding coefficients. The goal of training the SVM model is to find the SVs and their corresponding coefficients. During the test process, one can use the decision function depending on the SVs to discriminate the unlabeled data points. Consequently, the SVs play a core role for the SVM application. Many efforts have been made to enhance the computational efficiency of SVM. The related works can be roughly grouped into two categories, i.e., enhancing the training and test efficiencies. For example, in the literatures [10–14], the researchers devoted major efforts to efficiently training the SVM model, i.e., finding the SVs and the corresponding coefficients as quickly as possible.

Whereas, from an application point of view, the time-cost in the test phase of SVM is very pivotal for its practical application [15–20]. The test time-cost is linear with the number of SVs when implementing a well-trained SVM model, and a large number of SVs would result in increasing the time required to calculate the kernel matrix. In order to speed up the test process of SVM, many efforts have been made to condensing the solution of SVM, i.e., reducing the basis vectors used in its final decision function. Several representative methods are proposed in the past decades. In [21], Downs et al. developed an algorithm that leaves some SVs intact and removes some of the original SVs that are linearly dependent on the other SVs. In [22], the authors proposed a method to reduce the basis vectors by using smoothed separable case approximation. This method formulates the approximation of the SVM solution as a separable classification problem in the feature space, and then employs the hard-margin SVM to solve this problem. Both of these two methods remove some points from the support vector set of SVM, and so can speed up the test process.

* Corresponding author.
E-mail address: wxmwm@aliyun.com (X. Wang).

https://doi.org/10.1016/j.asoc.2020.107071
1568-4946/

Another class of methods can explicitly control the sparseness of the solution, i.e., the number of the basis vectors used in the decision function. In [23], Burges employed a linear combination of some basis vectors, which is much less than the SVs of SVM, approximates the normal vector of the decision hyperplane such that the distance to the original normal vector is minimized. In [24], the authors proposed to randomly select some points from the training data as the basis vectors, and then computes the coefficients according to the large margin principle. This method is named as reduced support vector machines (RSVM). The experimental results demonstrated that RSVM cannot guarantee that a good classification performance is offered when the number of the basis vectors is small [25]. The reason may be that the finally used basis vectors may not be good representatives of the training data since they are selected randomly. To deal with this issue, in [26], Keerthi et al. developed a greedy scheme that is directly related to the training cost to select the basis vectors for RSVM. In [27,28], Wu et al. directly learned a small number of basis vectors and the corresponding coefficients instead of the SVs yielded by SVM. Hu et al. [29] further extended the idea of [27,28] to learn a multiple kernel SVM. In essence, most existing methods employ the large margin principle, which is embodies in SVM, to obtain the basis vectors and the corresponding coefficients.

However, the major drawback of the conventional methods condensing the solution of SVM is that the employed criterions only try to maximize the margin between different classes. A key fact [30–33] should be noted that the upper bound of the generalization error of SVM depends on not only the margin between different classes but also the radius of the minimum enclosing hypersphere, which has the smallest radius among all hyperspheres enclosing the training data. In fact, the standard SVM can only exploit the margin and does not need to consider the radius. The reason is that the radius is fixed (a constant) since the used training data remain unchangeable when using the standard SVM. However, in the context of condensing the solution of SVM, the radius of the minimum enclosing hypersphere is not a constant but a function of the used expansion vectors. Section 3.1 will discuss this point in detail.

Inspiring by the works [30–33], we propose a novel method called sparse support vector machine guided by radius-margin bound (RMB-SSVM) to condensing the solution of SVM. The key insight behind RMB-SSVM is that the SVM model with the condensed solution is equivalent to constructing it in the transformation space determined by the basis vectors, and so the radius-margin bound changes with them. Based on this observation, RMB-SSVM employs a criterion that considers integratedly the margin between different classes and the radius of the minimum enclosing hypersphere to guide the selecting of the basis vectors and the learning of the corresponding coefficients. Moreover, as in [26], the sparseness of the resulting classifier can be explicitly controlled by manually setting the number of the used basis vectors. In the paper, we define the optimization model of RMB-SSVM and develop an alternate optimization strategy to deal with it. We also propose a heuristic scheme to select the basis vectors for the decision function. Further, we explore how to use the maximum pairwise distance over the pairs of data points to approximate the radius in our methodology. This can simplify the proposed model and reduce the model training time due to dodging a quadratic optimization problem used to compute the radius at each iteration. Finally, the comprehensive experiments are conducted, and the results show that the proposed methods are effective and can achieve a superior performance over the counterparts due to the key fact that it employs the radius-margin bound, which is directly related with the generalization ability of SVM, to guide the selecting of the basis vectors and the learning of the corresponding coefficients.

The rest of this paper is organized as follows. Section 2 briefly introduces the related work. Section 3 first discusses our motivation, and then defines the optimization model of the proposed method and details how to deal with it. Section 4 further explores how to approximate the radius in our methodology. The experimental results are reported in Section 5. Finally, Section 6 concludes the paper.

2. Related work

In the paper, we consider a given binary classification problem with a labeled training dataset $\{(\mathbf{x}_i, y_i) \,|i \in \mathcal{I}\}$ that contains N data points, where $\mathcal{I} = \{1, \ldots, N\}$, $\mathbf{x}_i \in \mathbb{R}^D$ represents the ith training data point and $y_i \in \{1, -1\}$ is its corresponding label. Here, D refers to the dimensionality of the data space.

2.1. Support vector machines

The traditional classification methods generally embody the empirical risk minimization principle. In other words, they classify the test data through some decision rules that are built by minimizing the sum of the misclassification errors of the training data. However, the SVM classifier has a solid theoretical foundation and stems from the structural risk minimization (SRM) principle [1,2]. The SRM principle is related to the probability of incorrectly classifying the unknown data points and suggests that the classifier should maximize the margin between different classes and simultaneously shrink the radius of the minimum enclosing hypersphere. Note, the radius is fixed since the training data remains unchangeable in the original data space. Therefore, in the separable case, SVM only maximizes the margin and defines its primal optimization problem with the hard margin as

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \\ &s.t.\ y_i\left(\mathbf{w}^T\varphi\left(\mathbf{x}_i\right)+b\right) \geq 1, i \in \mathcal{I} \end{aligned} \tag{1}$$

where $\varphi(\mathbf{x})$ represents a point given by an implicit mapping function that maps a sample in the original space into the feature space. In (1), the margin between different classes is characterized by $1/\|\mathbf{w}\|$. In order to handle the non-separable case, SVM with the ℓ_1-norm soft margin (denoted by ℓ_1-SVM) defines the following optimization problem

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

Here $\xi_i (i \in \mathcal{I})$ represents the negative slack variable for the ith data point, and C serves as a trade-off between the sum $\sum_{i\in\mathcal{I}}\xi_i$ of training error and the regularization $\|\mathbf{w}\|^2$ and is called as the regularization (trade-off) parameter.

Generally, the problem (2) is solved through its dual optimization problem [34], which is formulated as

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

where $k\left(\mathbf{x}_i,\mathbf{x}_j\right) = \varphi^T\left(\mathbf{x}_i\right)\varphi\left(\mathbf{x}_j\right)$ and it is called as kernel function. Suppose that $\boldsymbol{\alpha}^* = \left[\alpha_1^* \cdots \alpha_N^*\right]^T$ is the optimal solution to (2), we can compute the optimal bias b^* of the decision hyperplane according to the KKT conditions [34]. A data point $\mathbf{x}_i$ such that $\alpha_i^* > 0$ is called as support vector. We denote the number of the

support vectors (SVs) by N_{SV}. Thus, the final decision function of SVM can be formulated as

$$f(\mathbf{x}) = sign\left(\left(\mathbf{w}^*\right)^T \varphi(\mathbf{x}) + b^*\right) = sign\left(\sum_{i=1}^{N_{SV}} \alpha_i^* k(\mathbf{x}_i, \mathbf{x}) \varphi(\mathbf{x}_i) + b^*\right) \tag{4}$$

where

$$\mathbf{w}^* = \sum_{i=1}^{N_{SV}} \alpha_i^* \varphi(\mathbf{x}_i) \tag{5}$$

Obviously, the time-cost in the test phase of SVM is proportional to the number N_{SV} of the SVs, and would be expensive if N_{SV} was large.

For SVM with the hard margin, it was shown by Vapnik et al. [30] that $LOO\,Error \leq R^2\|\mathbf{w}\|^2/N$ holds, where *LOO Error* represents the leave-one-out (LOO) error, $\mathbf{w}$ is the optimal solution of the problem (1), and R is the radius of the minimum hypersphere enclosing the training data. The term $R^2\|\mathbf{w}\|^2$ is usually referred to as radius-margin bound (RMB). Obviously, the RMB is related with the upper bound of the *LOO Error* of SVM, and so characterizes the generalization ability of SVM. The RMB is generally used to conduct the hyperparameters selecting for SVM and it generally works quite well [30–33]. In [35], in order to conduct model select for ℓ_1-SVM, the authors explored several heuristic criterions and finally suggested the bound $\left(R^2 + \Delta/C\right)\left(\|\mathbf{w}\|^2/2 + C\sum_{i\in\mathcal{I}}\xi_i\right)$ for ℓ_1-SVM. The experimental results have demonstrated that this bound performs very well when being employed to conduct model selection for ℓ_1-SVM. Also, another advantage is that it is differentiable.

2.2. Reduced support vector machines

In order to speed up the test process of SVM, it is a natural way to reduce the used basis vectors (i.e. support vectors) in the function decision (4) since the test time-cost is proportional to the number of the basis vectors. For example, in RSVM [24], a random subset $\mathcal{Z} = \left\{\mathbf{z}_1, \ldots, \mathbf{z}_{N_{BV}}\right\}$ of the training data is first selected to be the basis vectors, and then the corresponding coefficients β_i $(i = 1, \ldots, N_{BV})$ are computed according to the large margin principle, which is embodied in SVM. More specifically, RSVM reformulates the norm vector $\mathbf{w}$ of the hyperplane in the feature space as follows

$$\mathbf{w} = \sum_{i=1}^{N_{BV}} \beta_i \varphi(\mathbf{z}_i) \tag{6}$$

Then, to obtain the optimal coefficients and the bias of the decision hyperplane, through substituting (6) into the objective function and the inequality constraint condition of (1), RSVM defines the primal optimization problem as follows

$$\begin{aligned} &\min_{\boldsymbol{\beta},b} \frac{1}{2}\boldsymbol{\beta}^T\mathbf{K}_{\mathcal{Z}}\boldsymbol{\beta} \\ &s.t.\ y_i\left(\boldsymbol{\beta}^T\varphi_{\mathcal{Z}}(\mathbf{x}_i) + b\right) \geq 1, i \in \mathcal{I} \end{aligned} \tag{7}$$

where $\varphi_{\mathcal{Z}}(\mathbf{x}_i) = \left[k(\mathbf{x}_i, \mathbf{z}_1), \ldots, k\left(\mathbf{x}_i, \mathbf{z}_{N_{BV}}\right)\right]^T$ and $\mathbf{K}_{\mathcal{Z}} = \left(k\left(\mathbf{z}_i, \mathbf{z}_j\right)\right)_{N_{BV}\times N_{BV}}$. Suppose that $\{\boldsymbol{\beta}^*, b^*\}$ solves the optimization problem (7), we can obtain the decision function of RSVM as follows

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{N_{BV}} \beta_i^* k(\mathbf{x}, \mathbf{z}_i) + b^*\right) \tag{8}$$

Obviously, if $N_{BV} \ll N_{SV}$, i.e., the number N_{EV} of the selected basis vectors is much smaller the number N_{SV} of the SVs yielded by directly using SVM, RSVM would take much less time in the test process in contrast with SVM.

The experimental results demonstrated that RSVM cannot guarantee that a good classification performance is offered when the number of the basis vectors is small [25]. The reason may be that the learned decision function heavily depends on the randomly selected subset $\mathcal{Z} = \left\{\mathbf{z}_1, \ldots, \mathbf{z}_{N_{BV}}\right\}$. In other words, the learned decision function would change if the selected basis vectors changed.

Different from RSVM, which randomly selects a subset of the training data as the basis vectors and then only optimizes the corresponding coefficients, SpSVM [26] and SLMC [27] learn the finally used basis vectors by different strategies. SpSVM starts with an empty set of basis functions and greedily chooses new basis vectors (from the training set) to improve the primal objective function. SLMC does not restrict the basis vectors to be a subset of the training data and iteratively optimizes the used basis vectors and the corresponding coefficients by a two-step alternate optimization strategy. However, both of them employ the margin between different classes to guide the learning of the basis vectors and the corresponding coefficients, and ignore the key fact that condensing the solution of SVM is equivalent to constructing the SVM model in the transformation space determined by the basis vectors. Thus, the radius-margin bound, which is related with the generalization ability of SVM, changes with them.

3. Sparse support vector machine guided by radius-margin bound

In the section, we first analyze the drawback associated with the conventional methods and our motivation, and then construct the optimization model of the proposed method and detail how to solve it.

3.1. Motivation and model

For the sake of clarity, we set

$$\mathbf{v} = \boldsymbol{\beta}\mathbf{K}_{\mathcal{Z}}^{0.5} \tag{9}$$

Thus, the problem (7) can be reformulated as follows

$$\begin{aligned} &\min_{\mathbf{v},b} \frac{1}{2}\|\mathbf{v}\|^2 \\ &s.t.\ y_i\left(\mathbf{v}^T\pi_{\mathcal{Z}}(\mathbf{x}_i) + b\right) \geq 1, i \in \mathcal{I} \end{aligned} \tag{10}$$

where $\pi_{\mathcal{Z}}(\mathbf{x}_i) = \mathbf{K}_{\mathcal{Z}}^{-0.5}\phi_{\mathcal{Z}}(\mathbf{x}_i) = \left[\mathbf{K}_{\mathcal{Z}}^{-0.5}k(\mathbf{x}_i, \mathbf{z}_1), \ldots, \mathbf{K}_{\mathcal{Z}}^{-0.5}k\left(\mathbf{x}_i, \mathbf{z}_{N_{BV}}\right)\right]^T$. Suppose that $\{\mathbf{v}, b\}$ solve the problem (10), we can obtain the coefficients of the basis vectors according to (9), i.e. $\boldsymbol{\beta} = \mathbf{K}_{\mathcal{Z}}^{-0.5}\mathbf{v}$. Note, the problem (10) embodies the large margin principle like the primal problem (1) of SVM. Here, the used dataset is $\{(\pi_{\mathcal{Z}}(\mathbf{x}_i), y_i)\,|i \in \mathcal{I}\}$, which lies in the transformation space determined by the basis vectors $\left\{\mathbf{z}_1, \ldots, \mathbf{z}_{N_{BV}}\right\}$ since each data point $\pi_{\mathcal{Z}}(\mathbf{x}_i)$ depends on them. Therefore, condensing the solution of SVM is equivalent to constructing the SVM model in the transformation space determined by the basis vectors. This throws a novel insight into the essence of condensing the solution of SVM. Most method condensing the solution of SVM differ in the manner by which the basis vectors are provided.

Note, the transformation space is only determined by the basis vectors. In other words, if the used basis vectors change, the transformation space would change accordingly. As a result, the margin between different classes and the radius of the minimum hypersphere in the transformation space change with the basis vectors. On the other hand, according to [30–33], the generalization performance of the large margin classifier is depend on not only the margin but also the radius. Therefore, the major

drawback of the existing methods condensing the solution of SVM is that they only consider the margin between different classes and fail to exploit the radius of the minimum enclosing hypersphere during selecting the basis vectors and learning the corresponding coefficients.

Inspired by the works [30–33], we propose a novel method called sparse support vector machine guided by radius-margin bound (RMB-SSVM) to condense the solution of SVM, i.e., select the used basis vectors and learn the corresponding coefficient in the decision function (8). In the separable case, RMB-SSVM defines the following model

$$\min_{\mathbf{w},b,\mathbf{a},R,\mathcal{Z}} \frac{1}{2}R^2\|\mathbf{w}\|^2 \quad s.t. \begin{cases} y_i\left(\mathbf{w}^T\varphi\left(\mathbf{x}_i\right)+b\right)\geq 1, i\in\mathcal{I} \\ \|\pi_{\mathcal{Z}}\left(\mathbf{x}_i\right)-\mathbf{a}\|^2\leq R^2, i\in\mathcal{I} \end{cases} \tag{11}$$

where $\mathbf{w}$ is described as (6), i.e. $\mathbf{w}=\sum_{i=1}^{N_{BV}}\beta_i\varphi\left(\mathbf{z}_i\right)$. By substituting (6) into (11) and setting $\mathbf{v}=\mathrm{K}_{\mathcal{Z}}^{0.5}\boldsymbol{\beta}$ as (9), the problem (11) can be further reformulated as follows

$$\min_{\mathbf{v},b,\mathbf{a},R,\mathcal{Z}} \frac{1}{2}R^2\|\mathbf{v}\|^2 \quad s.t. \begin{cases} y_i\left(\mathbf{v}^T\pi_{\mathcal{Z}}\left(\mathbf{x}_i\right)+b\right)\geq 1, i\in\mathcal{I} \\ \|\pi_{\mathcal{Z}}\left(\mathbf{x}_i\right)-\mathbf{a}\|^2\leq R^2, i\in\mathcal{I} \end{cases} \tag{12}$$

Obviously, in contrast with the other methods condensing the solution of SVM, the proposed method integratedly exploits the radius of the minimum enclosing hypersphere and the margin between different classes in the transformation space. Actually, the used decision function is parameterized by $\{\mathbf{Z},\boldsymbol{\beta},b\}$. The key insight is that the basis vectors and the corresponding coefficients work essentially as the hyperparameters of SVM. Therefore, we can borrow the basic idea of [30–33] to guide the selecting of the basis vectors and the learning of the corresponding coefficients.

Further, in order to deal with the non-separable case, by following the basic idea of [35], we define the following problem

$$\min_{\mathbf{v},b,\boldsymbol{\xi},\mathbf{a},R,\mathcal{Z}} \left(R^2+\frac{\Delta}{C}\right)\left(\frac{1}{2}\|\mathbf{v}\|^2+C\sum_{i\in\mathcal{I}}\xi_i\right) \quad s.t. \begin{cases} y_i\left(\mathbf{v}^T\pi_{\mathcal{Z}}\left(\mathbf{x}_i\right)+b\right)\geq 1-\xi_i, \xi_i\geq 0, \forall i\in\mathcal{I} \\ \|\pi_{\mathcal{Z}}\left(\mathbf{x}_i\right)-\mathbf{a}\|^2\leq R^2, \forall i\in\mathcal{I} \end{cases} \tag{13}$$

Here $\boldsymbol{\xi}=[\xi_1,\ldots,\xi_N]$ and Δ is a parameter and generally can be set $\Delta=1$ according to [35]. Suppose that $\{\mathbf{v}^*,b^*,\boldsymbol{\xi}^*,\mathbf{a}^*,R^*,\mathcal{Z}^*\}$ solves the problem (13), according to (9), we can compute the optimal coefficient vector $\boldsymbol{\beta}^*$ of the basis vectors by $\boldsymbol{\beta}^*=\mathrm{K}_{\mathcal{Z}^*}^{-0.5}\mathbf{v}^*$. Finally, we construct the decision function (8) by using $\{\boldsymbol{\beta}^*,b^*,\mathcal{Z}^*\}$. So, the key task is to how to select the basis vectors $\left\{\mathbf{z}_1,\ldots,\mathbf{z}_{N_{BV}}\right\}$ and compute the corresponding coefficients such that the objective function of (13) is minimum.

However, it is very difficult to directly find the global optimal solution of the optimization problem (13). Generally, we can employ a two-step alternate iteration optimization strategy to seek a local optimal. In this strategy, each iteration consists of two alternate steps. In the first step, we first keep the basis vectors $\left\{\mathbf{z}_1,\ldots,\mathbf{z}_{N_{BV}}\right\}$ fixed and solve the problem (13) with respect to $\{\mathbf{v},b,\boldsymbol{\xi},\mathbf{a},R\}$. In the second step, we optimize the basis vectors by the proposed heuristic scheme for the selecting of the basis vectors. In the paper, the used basis vectors are selected from the training data points.

3.2. Solving the optimization model when fixing the basis vectors

First, we should note that the terms in the parentheses of (13) are positive. Besides, in the problem (13), the first constraint does not depend on $\{\mathbf{a},R\}$ and the last one does not depend on $\{\mathbf{v},b,\boldsymbol{\xi}\}$. Thus, in the case of the basis vectors $\left\{\mathbf{z}_1,\ldots,\mathbf{z}_{N_{BV}}\right\}$ being given, the formulation (13) reduces to

$$\mathcal{J}\left(\mathcal{Z}\right)=\mathcal{J}_1\left(\mathcal{Z}\right)\times\mathcal{J}_2\left(\mathcal{Z}\right) \tag{14}$$

where

$$\mathcal{J}_1\left(\mathcal{Z}\right)=\begin{cases}\min\limits_{\mathbf{a},R} R^2+\dfrac{\Delta}{C} \\ s.t.\|\pi_{\mathcal{Z}}\left(\mathbf{x}_i\right)-\mathbf{a}\|^2\leq R^2, \forall i\in\mathcal{I}\end{cases} \tag{15}$$

and

$$\mathcal{J}_2\left(\mathcal{Z}\right)=\begin{cases}\min\limits_{\mathbf{v},b,\boldsymbol{\xi}} \dfrac{1}{2}\|\mathbf{v}\|^2+C\sum\limits_{i\in\mathcal{I}}\xi_i \\ s.t.\ y_i\left(\mathbf{v}^T\pi_{\mathcal{Z}}\left(\mathbf{x}_i\right)+b\right)\geq 1-\xi_i, \xi_i\geq 0, \forall i\in\mathcal{I}\end{cases} \tag{16}$$

Obviously, in order to calculate the objective value $\mathcal{J}\left(\mathcal{Z}\right)$ in the case of a fixed $\mathcal{Z}$, we need to first calculate the objective value $\mathcal{J}_1\left(\mathcal{Z}\right)$ and the objective value $\mathcal{J}_2\left(\mathcal{Z}\right)$, which are respectively the optimal objective values of one support vector data description (SVDD) problem [36] and one ℓ_1-SVM problem. Here, note that the objective function of (15) adds an extra constant term Δ/C in contrast with one of SVDD. By using the Wolfe dual theory [34], the above problems (16) and (17) can be respectively written as follows

$$\mathcal{J}_1\left(\mathcal{Z}\right)=\begin{cases}\max\limits_{\boldsymbol{\gamma}} \sum\limits_{i\in\mathcal{I}}\gamma_i k_{\mathcal{Z}}\left(\mathbf{x}_i,\mathbf{x}_i\right)-\sum\limits_{i\in\mathcal{I}}\sum\limits_{j\in\mathcal{I}}\gamma_i\gamma_j k_{\mathcal{Z}}\left(\mathbf{x}_i,\mathbf{x}_j\right)+\dfrac{\Delta}{C} \\ s.t.\ \sum\limits_{i\in\mathcal{I}}\gamma_i=1, 0\leq\gamma_i\leq 1, \forall i\in\mathcal{I}\end{cases} \tag{17}$$

and

$$\mathcal{J}_2\left(\mathcal{Z}\right)=\begin{cases}\max\limits_{\boldsymbol{\alpha}} \sum\limits_{i\in\mathcal{I}}\alpha_i-\dfrac{1}{2}\sum\limits_{i\in\mathcal{I}}\sum\limits_{j\in\mathcal{I}}\alpha_i\alpha_j y_i y_j k_{\mathcal{Z}}\left(\mathbf{x}_i,\mathbf{x}_j\right) \\ s.t.\ \sum\limits_{i\in\mathcal{I}}y_i\alpha_i=0, 0\leq\alpha_i\leq C, \forall i\in\mathcal{I}\end{cases} \tag{18}$$

where

$$k_{\mathcal{Z}}\left(\mathbf{x}_i,\mathbf{x}_j\right)=\pi_{\mathcal{Z}}^T\left(\mathbf{x}_i\right)\pi_{\mathcal{Z}}\left(\mathbf{x}_j\right)=\varphi_{\mathcal{Z}}^T\left(\mathbf{x}_i\right)\mathrm{K}_{\mathcal{Z}}^{-1}\varphi_{\mathcal{Z}}\left(\mathbf{x}_j\right) \tag{19}$$

Thus, after respectively solving the two quadratic optimization problems which correspond to the dual forms of SVDD and one of SVM, we can obtain the objective value $\mathcal{J}\left(\mathcal{Z}\right)$ of (13) for a fixed $\mathcal{Z}$ with (14).

3.3. Updating the basis vectors

In this subsection, we will discuss in detail how to update the basis vectors $\left\{\mathbf{z}_1,\ldots,\mathbf{z}_{N_{BV}}\right\}$. For a basis vector $\mathbf{z}_u$ in the basis vector set $\mathcal{Z}$, our strategy is to select a vector from the candidate set to substitute it such that the objective function value would decrease as much as possible. The candidate set contains the data points belonging to the training set but not to the basis vector set $\mathcal{Z}$. So, there exists a key problem: for each vector in the basis vector set, how to choose a suitable vector from the candidate set to substitute it.

3.3.1. Strategy of updating the basis vectors

We assume that a basis vector $\mathbf{z}_u$ would been removed from the basis vector set $\mathcal{Z}$. After removing it, in order to remain the cardinality of the basis vector set unchangeable, we need to select a vector from the candidate set to add to the basis vector set $\mathcal{Z}$. We denote the selected vector from the candidate set by $\mathbf{z}_s$.

A feasible $\mathbf{z}_s$, which is used as a substitute for $\mathbf{z}_u$ in the basis vector set $\mathcal{Z}$, must make the objective function value decrease.

In particular, in the case of fixing all other related variables, we denote the partial derivative of $J(\mathcal{Z})$ with respect to $\varphi(\mathbf{z}_u)$ by

$$\nabla_{\varphi(\mathbf{z}_u)} = \frac{\partial J(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} \tag{20}$$

Obviously, $\nabla_{\varphi(\mathbf{z}_u)}$ indicates the direction along which the objective function value increases most rapidly. Suppose that $\varphi(\mathbf{z}_s)$ is the updated point corresponding to $\varphi(\mathbf{z}_u)$ in the feature space by the gradient-descent method [34], i.e., the following holds

$$\varphi(\mathbf{z}_s) = \varphi(\mathbf{z}_u) - \lambda \nabla_{\varphi(\mathbf{z}_u)} \tag{21}$$

Here, λ is the step size in the gradient-descent method. Then, we can use $\mathbf{z}_s$ as a substitute for $\mathbf{z}_r$ in the basis vector set. Further, we denote $\nabla_s = \varphi(\mathbf{z}_u) - \varphi(\mathbf{z}_s)$ and have

$$\nabla_s = \varphi(\mathbf{z}_u) - \varphi(\mathbf{z}_s) = \lambda \nabla_{\varphi(\mathbf{z}_u)} \tag{22}$$

The above equation shows that $\nabla_s = \varphi(\mathbf{z}_u) - \varphi(\mathbf{z}_s)$ should be linear with the gradient $\nabla_{\varphi(\mathbf{z}_u)}$.

Based on the above discussion and inspired by [37], in order to update the point $\mathbf{z}_u$ in the basis vector set $\mathcal{Z}$, our strategy is to select a data point $\mathbf{z}_s$ over the candidate set such that the vector $\nabla_s = \varphi(\mathbf{z}_u) - \varphi(\mathbf{z}_s)$ is most collinear with the gradient $\nabla_{\varphi(\mathbf{z}_u)}$ of the objective function with respect to $\varphi(\mathbf{z}_u)$. To be specific, we can find the index of $\mathbf{z}_s$ from the candidate set by solving the following problem

$$\begin{aligned} s &= \underset{k \in \mathcal{I}, k \notin \mathcal{I}_{BV}}{\operatorname{argmax}} \left(\frac{\left(\left(\nabla_{\varphi(\mathbf{z}_u)} \right)^T \nabla_k \right)^2}{\left(\nabla_{\varphi(\mathbf{z}_u)} \right)^T \nabla_k} \right) \\ &= \underset{k \in \mathcal{I}, k \notin \mathcal{I}_{BV}}{\operatorname{argmax}} \left(\frac{\left(\left(\nabla_{\varphi(\mathbf{z}_u)} \right)^T (\varphi(\mathbf{z}_u) - \varphi(\mathbf{x}_k)) \right)^2}{(\varphi(\mathbf{z}_u) - \varphi(\mathbf{x}_k))^T (\varphi(\mathbf{z}_u) - \varphi(\mathbf{x}_k))} \right) \end{aligned} \tag{23}$$

Here $\nabla_k = \varphi(\mathbf{z}_u) - \varphi(\mathbf{z}_k)$, and $\mathcal{I}_{BV}$ is the index set of the basis vectors in the training set and $\mathcal{I}_{BV} \subset \mathcal{I}$.

3.3.2. Computing the inner product between the gradient and a vector

According to the above discussion, in order to select a data point from the candidate set, we should solve the problem (23). This seemingly needs to first compute $\nabla_{\varphi(\mathbf{z}_u)}$ according to (20). Actually, it is not necessary and even impossible in that the feature space might be infinite dimensional. Note, our goal is to find $\mathbf{z}_s$ through solving the problem (23), where the key operation is the inner product between the gradient $\nabla_{\varphi(\mathbf{z}_u)}$ and a vector $\phi(\mathbf{x}_k)$ in the feature space. So, in the following, we will detail how to compute the inner product.

First, according to (20), the inner product between $\partial \mathcal{J}_1(\mathcal{Z}) / \partial \varphi(\mathbf{z}_u)$ and the vector $\varphi(\mathbf{x}_k)$ is

$$\begin{aligned} &\left(\frac{\partial \mathcal{J}_1(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k) \\ &= \left(\frac{\partial \left(\sum_{i \in \mathcal{I}} \gamma_i k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \gamma_i \gamma_j k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_j) + \frac{A}{C} \right)}{\partial \varphi(\mathbf{z}_u)} \right)^T \\ &\quad \times \varphi(\mathbf{x}_k) \\ &= \sum_{i \in \mathcal{I}} \gamma_i \left(\frac{\partial k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_i)}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k) - \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \gamma_i \gamma_j \left(\frac{\partial k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \varphi(\mathbf{z}_u)} \right)^T \\ &\quad \times \varphi(\mathbf{x}_k) \end{aligned} \tag{24}$$

Analogously, for $\mathcal{J}_2(\mathcal{Z})$ of (18), we have

$$\begin{aligned} &\left(\frac{\partial \mathcal{J}_2(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k) \\ &= \left(\frac{\partial \left(\sum_{i \in \mathcal{I}} \alpha_i - \frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \alpha_i \alpha_j y_i y_j k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_j) \right)}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k) \\ &= -\frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \alpha_i \alpha_j y_i y_j \left(\frac{\partial k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k) \end{aligned} \tag{25}$$

In (27) and (28), the key operation is to compute the inner product $\left(\partial k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_j) / \partial \varphi(\mathbf{z}_u) \right)^T \varphi(\mathbf{x}_k)$. We detail how to calculate it in Appendix.

Then, according to (14), the inner product between the derivative $\nabla_{\varphi(\mathbf{z}_u)}$ and a vector $\varphi(\mathbf{x}_k)$ in the feature space can be expressed as

$$\begin{aligned} &\left(\nabla_{\varphi(\mathbf{z}_r)} \right)^T \varphi(\mathbf{x}_k) \\ &= \left(\frac{\partial \mathcal{J}(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k) \\ &= \left(\mathcal{J}_2(\mathcal{Z}) \frac{\mathcal{J}_1(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} + \mathcal{J}_1(\mathcal{Z}) \frac{\mathcal{J}_2(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k) \\ &= \mathcal{J}_2(\mathcal{Z}) \left(\left(\frac{\mathcal{J}_1(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k) \right) + \mathcal{J}_1(\mathcal{Z}) \left(\left(\frac{\mathcal{J}_2(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k) \right) \end{aligned} \tag{26}$$

Finally, according to the above discussion, we outline the main algorithmic steps of updating the basis vectors in **Algorithm 1**.

Algorithm 1 Updating the basis vector set

Input:	The old basis vectors $\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\}$, the optimal solutions to (17) and (18);
Output:	The updated basis vectors $\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\}$
Step 1:	**For** $u = 1, \cdots, N_{BV}$
Step 2:	**Compute** $\left(\frac{\partial \mathcal{J}_1(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k)$ and $\left(\frac{\partial \mathcal{J}_2(\mathcal{Z})}{\partial \varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k)$ with (27) and (28), respectively;
Step 3:	**Compute** $\left(\nabla_{\varphi(\mathbf{z}_u)} \right)^T \varphi(\mathbf{x}_k)$ with (29);
Step 4:	**Find** the vector $\mathbf{z}_s$ with (23);
Step 6:	**Remove** the basis vector $\mathbf{z}_u$ from the basis vector set and **add** $\mathbf{z}_s$ to the basis vector set if the vector $\mathbf{z}_s$ could make the objective function value decrease;
Step 7:	**End for**
Step 8:	**Return** the updated basis vectors $\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\}$.

3.4. Algorithm

According to the above discussion, we outline the main algorithmic steps of RMB-SSVM in Algorithm 2. Note, there is no guarantee that this procedure can offer the global optimum and the initial values of $\mathcal{Z}$ might impact the solution quality. However, this procedure usually can provide a satisfying solution.

Once finishing the alternate iteration optimization procedure for the optimization model (13), we need to construct the decision function (8). Suppose that **Algorithm 2** gives the optimized

basis vector set $\mathcal{Z}^* = \left\{\mathbf{z}_i^*, \ldots, \mathbf{z}_{N_{BV}}^*\right\}$ and the finally obtained Lagrange multipliers $\left\{\alpha_1^*, \ldots, \alpha_N^*\right\}$. Thus, we can compute the optimal threshold b^* according to the KKT conditions [34] as in SVM. Further, according to (9), we can the optimal basis coefficient vector $\boldsymbol{\beta}^*$ as follows

$$\boldsymbol{\beta}^* = \mathbf{v}^* \mathbf{K}_{\mathcal{Z}^*}^{-0.5} = \sum_{i \in \mathcal{I}} \alpha_i^* y_i \pi_{\mathcal{Z}^*}(\mathbf{x}_i) \mathbf{K}_{\mathcal{Z}^*}^{-0.5} \tag{27}$$

where $\mathbf{v}^* = \sum_{i \in \mathcal{I}} \alpha_i^* y_i \pi_{\mathcal{Z}^*}(\mathbf{x}_i)$ and $\mathbf{K}_{\mathcal{Z}^*} = \left(k\left(\mathbf{z}_i^*, \mathbf{z}_j^*\right)\right)_{N_{BV} \times N_{BV}}$. Finally, we can obtain the decision function of RMB-SSVM as follows

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{N_{BV}} \beta_i^* k\left(\mathbf{x}, \mathbf{z}_i^*\right) + b^*\right) \tag{28}$$

Algorithm 2 Spare support vector machine guided by radius-margin bound (RMB-SSVM)

Input: The training dataset $\{(\mathbf{x}_i, y_i) | i \in \mathcal{I}\}$, the number N_{BV} of the basis vectors, the adopted kernel and the related parameter, the regularization parameter C and the parameter Δ;

Output: The optimized basis vectors $\left\{\mathbf{z}_i^*, \cdots, \mathbf{z}_{N_{BV}}^*\right\}$ and the finally obtained Lagrange multipliers $\left\{\alpha_1^*, \cdots, \alpha_N^*\right\}$;

Step 1: **Initialize** the basis vectors $\left\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\right\}$;

Step 2: **While** stopping criterion not met **do**

Step 3: **Compute** $\mathcal{J}_1(\mathcal{Z})$ and $\mathcal{J}_2(\mathcal{Z})$ with (17) and (18) in the case of fixing the basis vectors $\left\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\right\}$, respectively;

Step 4: **Compute** $\mathcal{J}(\mathcal{Z})$ with (14);

Step 5: **Update** the basis vectors $\left\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\right\}$ by Algorithm 1;

Step 6: **End while**

Step 7: **Return the solution:** the finally updated basis vector set $\left\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\right\}$ and the finally obtained Lagrange multipliers $\{\alpha_1, \cdots, \alpha_N\}$.

4. Alternative to the radius

In the above discussion, ones need to calculate the radius of the minimum enclosing hypersphere at each iteration, which needs to solve a quadratic optimization problem. In this section, we will explore how to approximate the radius in our methodology.

4.1. Relationship between the radius and the maximum pairwise distance

First, for the dataset $\{\pi_{\mathcal{Z}}(\mathbf{x}_i) | i \in \mathcal{I}\}$, the maximum pairwise distance is formulated as

$$\begin{aligned} D_{\mathcal{Z}} &= \sqrt{\max_{i \in \mathcal{I}, j \in \mathcal{I}} \|\pi_{\mathcal{Z}}(\mathbf{x}_i) - \pi_{\mathcal{Z}}\left(\mathbf{x}_j\right)\|^2} \\ &= \sqrt{\max_{i \in \mathcal{I}, j \in \mathcal{I}} \left(\|\pi_{\mathcal{Z}}(\mathbf{x}_i)\|^2 + \|\pi_{\mathcal{Z}}\left(\mathbf{x}_j\right)\|^2 - 2\pi_{\mathcal{Z}}^T(\mathbf{x}_i)\pi_{\mathcal{Z}}\left(\mathbf{x}_j\right)\right)} \\ &= \sqrt{\max_{i \in \mathcal{I}, j \in \mathcal{I}} \left(k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_i) + k_{\mathcal{Z}}\left(\mathbf{x}_j, \mathbf{x}_j\right) - 2k_{\mathcal{Z}}\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)} \end{aligned} \tag{29}$$

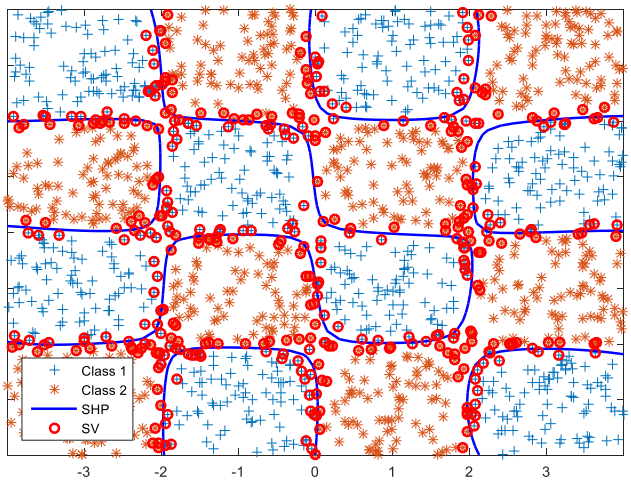


Fig. 1. The separating hyperplanes of ℓ_1-SVM on the synthetic dataset.

Further, according to [38], the following $D_{\mathcal{Z}}/2 \leq R \leq \left(1+\sqrt{3}\right) D_{\mathcal{Z}}/4$ holds. Here R refers to the radius of the minimum hypersphere enclosing the dataset $\{\pi_{\mathcal{Z}}(\mathbf{x}_i) | i \in \mathcal{I}\}$. So, we have

$$\left(4 - 2\sqrt{3}\right) R^2 \leq \frac{1}{4} D_{\mathcal{Z}}^2 \leq R^2 \tag{30}$$

The formula (35) reveals the key fact that there exists a close relationship between the radius and the maximum pairwise distance.

4.2. Model and solution

By using $D_{\mathcal{Z}}^2/4$ in (33) instead of R^2 in (13), we define the optimization model of spare support vector machine constrained by maximum pairwise distance (MPD-SSVM) as follows

$$\begin{aligned} &\min_{\mathbf{v}, b, \boldsymbol{\xi}, \mathcal{Z}} \left(\frac{1}{4} D_{\mathcal{Z}}^2 + \frac{\Delta}{C}\right)\left(\frac{1}{2}\|\mathbf{v}\|^2 + C\sum_{i \in \mathcal{I}} \xi_i\right) \\ &s.t.\ y_i\left(\mathbf{v}^T \pi_{\mathcal{Z}}(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in \mathcal{I} \end{aligned} \tag{31}$$

where

$$D_{\mathcal{Z}}^2 = \max_{i \in \mathcal{I}, j \in \mathcal{I}} \left(k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_i) + k_{\mathcal{Z}}\left(\mathbf{x}_j, \mathbf{x}_j\right) - 2k_{\mathcal{Z}}\left(\mathbf{x}_i, \mathbf{x}_j\right)\right) \tag{32}$$

Obviously, MPD-SSVM tries to maximize the margin between different classes while shrinking the region covering the training data points. Consequently, it complies with the basic idea of minimizing the radius-margin bound.

As has been discussed in Section 3, one way to solve the optimization problem (34) is to employ a two-step iterative strategy. In the first step, we will keep $\mathcal{Z}$ fixed. As a consequence, only one quadratic optimization problem, which responds to SVM, needs to be solved. Besides, we compute $D_{\mathcal{Z}}^2$ by using (35) instead of computing the radius R^2 of the minimum enclosing hypersphere. This dodges a quadratic optimization problem used to compute the radius at each iteration and can save the training time. Simultaneously, the modification does not severely impair the generalization performance according the experimental results. In the second step, we also optimize over $\mathcal{Z}$ by Algorithm 1. In the following, we will detail how to solve the problem (34).

For a fixed $\mathcal{Z}$, we denote the objective function of (34) by $\widehat{\mathcal{J}}(\mathcal{Z})$. Similarly to Section 3.2, for the problem (34) in the case of fixing $\mathcal{Z}$, we have

$$\widehat{\mathcal{J}}(\mathcal{Z}) = \widehat{\mathcal{J}}_1(\mathcal{Z}) \times \widehat{\mathcal{J}}_2(\mathcal{Z}) \tag{33}$$

where

$$\widehat{\mathcal{J}}_1(\mathcal{Z}) = \frac{1}{4}D_{\mathcal{Z}}^2 + \frac{\Delta}{C} \tag{34}$$

and

$$\widehat{\mathcal{J}}_2(\mathcal{Z}) = \begin{cases} \min\limits_{\mathbf{v},b,\xi} \dfrac{1}{2}\|\mathbf{v}\|^2 + C\sum\limits_{i\in\mathcal{I}}\xi_i \\ s.t.\ y_i\left(\mathbf{v}^T\pi_{\mathcal{Z}}(\mathbf{x}_i) + b\right) \geq 1-\xi_i, \xi_i \geq 0, \forall i \in \mathcal{I} \end{cases} = \mathcal{J}_2(\mathcal{Z}) \tag{35}$$

For (37), we can easily compute it. Here, we suppose that $\{\mathbf{x}_{i'}, \mathbf{x}_{j'}\}$ makes the following

$$k_{\mathcal{Z}}(\mathbf{x}_{i'}, \mathbf{x}_{i'}) + k_{\mathcal{Z}}(\mathbf{x}_{j'}, \mathbf{x}_{j'}) - 2k_{\mathcal{Z}}(\mathbf{x}_{i'}, \mathbf{x}_{j'}) = \max_{i\in\mathcal{I}, j\in\mathcal{I}} \left(k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_i) + k_{\mathcal{Z}}(\mathbf{x}_j, \mathbf{x}_j) - 2k_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}_j)\right)$$

holds. Therefore, we have

$$D_{\mathcal{Z}}^2 = k_{\mathcal{Z}}(\mathbf{x}_{i'}, \mathbf{x}_{i'}) + k_{\mathcal{Z}}(\mathbf{x}_{j'}, \mathbf{x}_{j'}) - 2k_{\mathcal{Z}}(\mathbf{x}_{i'}, \mathbf{x}_{j'}) \tag{36}$$

For (38), it is obviously equal to (16) in Section 3.2, and so we can solve it by using its duality (18). Consequently, it is clear that we do not need to compute the radius of MEB at each iteration since using the approximation $D_{\mathcal{Z}}^2$ which can be computed by (39). Thus, we can save the training time to some degree.

For $\widehat{\mathcal{J}}_1(\mathcal{Z})$, the inner product between $\partial\widehat{\mathcal{J}}_1(\mathcal{Z})/\partial\varphi(\mathbf{z}_u)$ and the vector $\varphi(\mathbf{x}_k)$ is

$$\begin{aligned} &\left(\frac{\partial\widehat{\mathcal{J}}_1(\mathcal{Z})}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k) \\ &= \left(\frac{\partial\left(\frac{1}{4}D_{\mathcal{Z}}^2 + \frac{\Delta}{C}\right)}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k) \\ &= \frac{1}{4}\left(\frac{\partial\left(k_{\mathcal{Z}}(\mathbf{x}_{i'}, \mathbf{x}_{i'}) + k_{\mathcal{Z}}(\mathbf{x}_{j'}, \mathbf{x}_{j'}) - 2k_{\mathcal{Z}}(\mathbf{x}_{i'}, \mathbf{x}_{j'})\right)}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k) \\ &= \frac{1}{4}\begin{pmatrix} \left(\frac{\partial k_{\mathcal{Z}}(\mathbf{x}_{i'}, \mathbf{x}_{i'})}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k) + \left(\frac{\partial k_{\mathcal{Z}}(\mathbf{x}_{j'}, \mathbf{x}_{j'})}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k) \\ -2\left(\frac{\partial k_{\mathcal{Z}}(\mathbf{x}_{i'}, \mathbf{x}_{j'})}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k) \end{pmatrix} \end{aligned} \tag{37}$$

Analogously, for $\widehat{\mathcal{J}}_2(\mathcal{Z})$, due to the fact $\widehat{\mathcal{J}}_2(\mathcal{Z}) = \mathcal{J}_2(\mathcal{Z})$, we have

$$\left(\frac{\partial\widehat{\mathcal{J}}_2(\mathcal{Z})}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k) = \left(\frac{\partial\mathcal{J}_2(\mathcal{Z})}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k) \tag{38}$$

Finally, according to (36), the inner product between the derivative $\partial\widehat{\mathcal{J}}(\mathcal{Z})/\partial\varphi(\mathbf{z}_u)$ and a vector $\varphi(\mathbf{x}_k)$ in the feature space can be expressed as

$$\begin{aligned} &\left(\frac{\partial\widehat{\mathcal{J}}(\mathcal{Z})}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k) \\ &= \widehat{\mathcal{J}}_2(\mathcal{Z})\left(\left(\frac{\mathcal{J}_1(\mathcal{Z})}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k)\right) + \widehat{\mathcal{J}}_1(\mathcal{Z})\left(\left(\frac{\mathcal{J}_2(\mathcal{Z})}{\partial\varphi(\mathbf{z}_u)}\right)^T\varphi(\mathbf{x}_k)\right) \end{aligned} \tag{39}$$

To summarize, the main algorithmic steps of MPD-SSVM is similar to Algorithm 2 and outlined in Algorithm 3. Once Algorithm 3 gives the optimized basis set $\mathcal{Z}^* = \{\mathbf{z}_i^*, \ldots, \mathbf{z}_{N_{BV}}^*\}$ and the finally obtained Lagrange multipliers $\{\alpha_1^*, \ldots, \alpha_N^*\}$, analogously to Section 3.4, we can compute the optimal threshold b^* according to the KKT conditions and the optimal basis coefficient vector $\boldsymbol{\beta}^*$ with (30). Finally, by using $\{\mathbf{z}_i^*, \ldots, \mathbf{z}_{N_{BV}}^*\}$, $\boldsymbol{\beta}^*$ and b^*, we can construct the final decision function of MPD-SSVM as (31).

Algorithm 3 Spare support vector machine constrained by maximum pairwise distance (MPD-SSVM)

Input:	The training dataset $\{(\mathbf{x}_i, y_i)\,|\,i \in \mathcal{I}\}$, the number N_{BV} of basis vectors, the adopted kernel and the related parameter, the regularization parameter C and the parameter Δ;
Output:	The optimized basis vectors $\{\mathbf{z}_i^*, \cdots, \mathbf{z}_{N_{BV}}^*\}$ and the finally obtained Lagrange multipliers $\{\alpha_1^*, \cdots, \alpha_N^*\}$;
Step 1:	**Initialize** the basis vectors $\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\}$;
Step 2:	**While** stopping criterion not met **do**
Step 3:	**Compute** $D_{\mathcal{Z}}^2$ with (39);
Step 4:	**Compute** $\widehat{\mathcal{J}}_1(\mathcal{Z})$ and $\widehat{\mathcal{J}}_2(\mathcal{Z})$ with (37) and (38) in the case of fixing the basis vectors $\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\}$, respectively;
Step 5:	**Compute** $\widehat{\mathcal{J}}(\mathcal{Z})$ with (36);
Step 6:	**Update** the basis vectors $\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\}$ by Algorithm 1;
Step 7:	**End while**
Step 8:	**Return the solution:** the finally updated basis vector set $\{\mathbf{z}_1, \cdots, \mathbf{z}_{N_{BV}}\}$ and the finally obtained Lagrange multipliers $\{\alpha_1, \cdots, \alpha_N\}$.

5. Experiment

In this section, we report the comprehensively experimental results of the proposed method. In the first experiment, we conduct the experiment on a synthetic dataset to demonstrate the effectiveness of the proposed method. In the second experiment, we experimentally investigate the influence of the related parameters on the performance of the proposed method. The third experiment evaluates the generalization performance of the proposed method by comparing it with other methods. Finally, we further test the proposed method on several large-scale datasets.

In the all experiments, we use the LIBSVM [39] package to solve the inner SVM and SVDD optimization problems.

5.1. Experimental evaluations on synthetic data

In this subsection, we conduct the experiment on a synthetic data to demonstrate the effectiveness of the proposed method. In this experiment, we create a checkerboard-shaped data in the two dimensional data space. This data consists of two classes and each class contains 800 data points. In the experiment, we adopted the RBF kernel, i.e. $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2\right)$. Besides, we set the kernel parameter σ and the regularization parameter C to 1 and 5, respectively.

First, we test the conventional method ℓ_1-SVM on this dataset. Fig. 1 shows the separating hyperplane given by ℓ_1-SVM. Here, SHP refers to the separating hyperplane and SV is the abbreviation for support vectors. In this dataset, ℓ_1-SVM yields 384 support vectors.

Fig. 2 shows the experimental results of SpSVM [26] under different numbers of the basis vectors. Here, the points marked

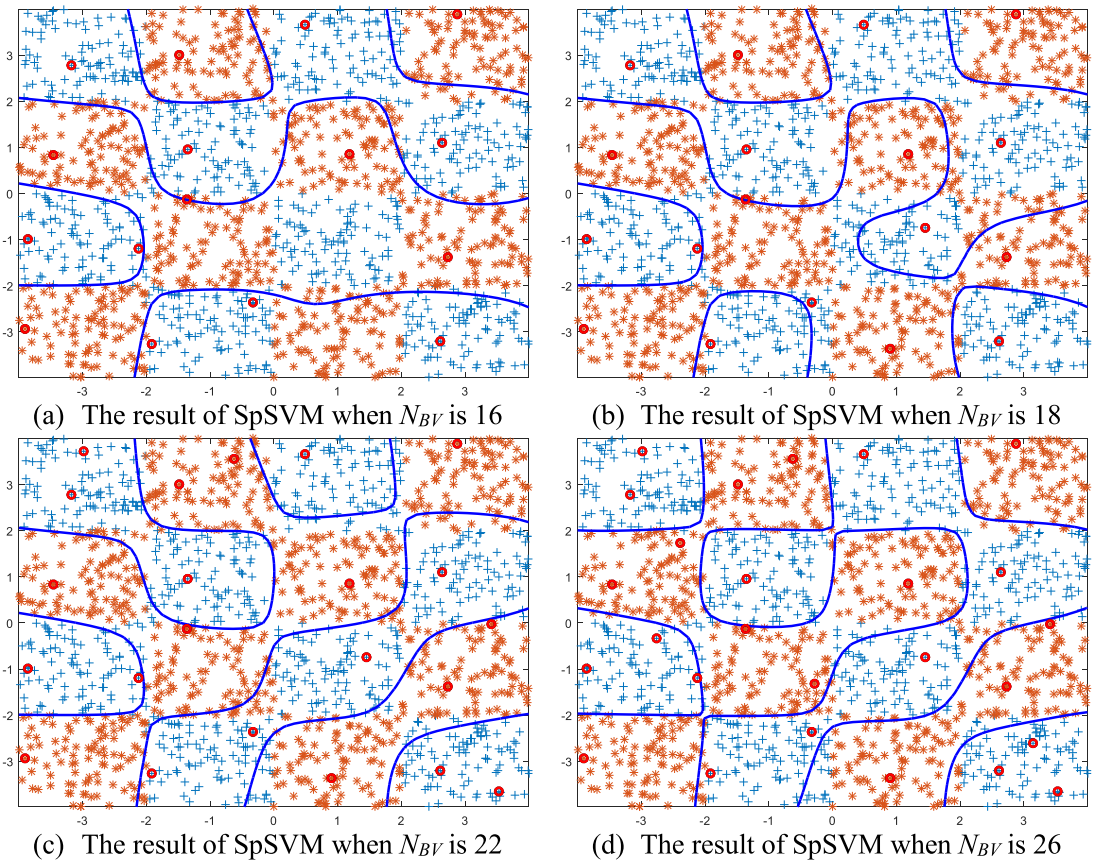


Fig. 2. The separating hyperplanes of SpSVM under different numbers of the basis vectors.

with red circle are the basis vectors given by SpSVM. According to Fig. 2(a), where the number N_{BV} of the basis vectors is set to 16, the separating hyperplane offered by SpSVM is quite poor. If we increase the used basis vectors, as is shown in Fig. 2(b) to (d), SpSVM cannot still obtain a satisfactory result. The reason is that SpSVM only embodies the large margin principle and employs a greedy scheme to select the used basis vector in the decision function.

Figs. 3 and 4 show the experimental results of the proposed methods, RMB-SSVM and MPD-SSVM, under different numbers of the basis vectors. The points marked with black circle represent the initial basis vectors, and ones with red circle are the basis vectors selected by the proposed methods. According to Fig. 3(a), where we use 16 basis vectors, RMB-SSVM yields an acceptable separating hyperplane. According to Fig. 3(b) to (d), the quality of the separating hyperplane of RMB-SSVM is further improved with the number N_{BV} of the basis vectors increasing. In Fig. 3(c) and (d), where only 22 and 26 basis vectors are used, the quality of the separating hyperplanes of RMB-SSVM is not inferior to one of SVM as in Fig. 1. Similar phenomenon can be observed in Fig. 4, which shows the experimental results of MPD-SSVM. This indicates that it is effective to approximating the radius of the minimum enclosing hypersphere by the maximum pairwise distance over the pairs of data points in our methodology.

To sum up, the experimental results show the effectiveness of the proposed methods, including RMB-SSVM and MPD-SSVM. They can generally yield an acceptable separating hyperplane by using the far fewer basis vectors than the support vectors of SVM. In contrast with SpSVM, RMB-SSVM and MPD-SSVM further improve the quality of the decision function in that they employ a criterion that is directly related to the generalization ability of the classifier.

5.2. Experimental investigations of parameters influence on the performance

According to (13) and (34), the proposed methods introduce the regularization parameter C and the parameter Δ. In this subsection, we will experimentally investigate the influence of the parameters on the performance.

In the experiment, we use the Pima and German datasets, which were selected from the UCI repository [40]. Pima and German consist of 768 data points with 8 features and 1000 data points with 24 features, respectively. Due to large value ranges of the features, we normalized each data by linearly scaling all features to the range [0,1] before conducting experiments. In the experiment, 60% data points of each dataset were randomly selected for training and the rest for testing. In the experiment, we adopted the RBF kernel, i.e. $\mathrm{k}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2\right)$.

First, we experimentally investigate the influences of the parameter Δ and the regularization parameter C on the proposed methods, RMB-SSVM and MPD-SSVM. We vary the regularization parameter C and the parameter Δ in the sets {0.01, 0.1, 1, 10, 100, 500, 1000, 5000, 10000} and the {0.05, 0.1, 0.5, 1, 2, 4, 8, 16}, respectively. Besides, the kernel parameter σ and the number of the basis vectors are set to 2 and 10. Fig. 5(a) and (b) show respectively the experimental results of RMB-SSVM and MPD-SSVM on the Pima dataset. These results demonstrate that the parameter Δ has less influence on the whole in contrast with the regularization parameter C. From Fig. 5(c) and (d), which shows the results on the German dataset, we can observe the similar phenomena. According to the results, when $\Delta = 1$, the accuracy under different regularization parameters is acceptable although it might not be the best. In order to obtain a better performance, the regularization parameter C tends to be a large value.

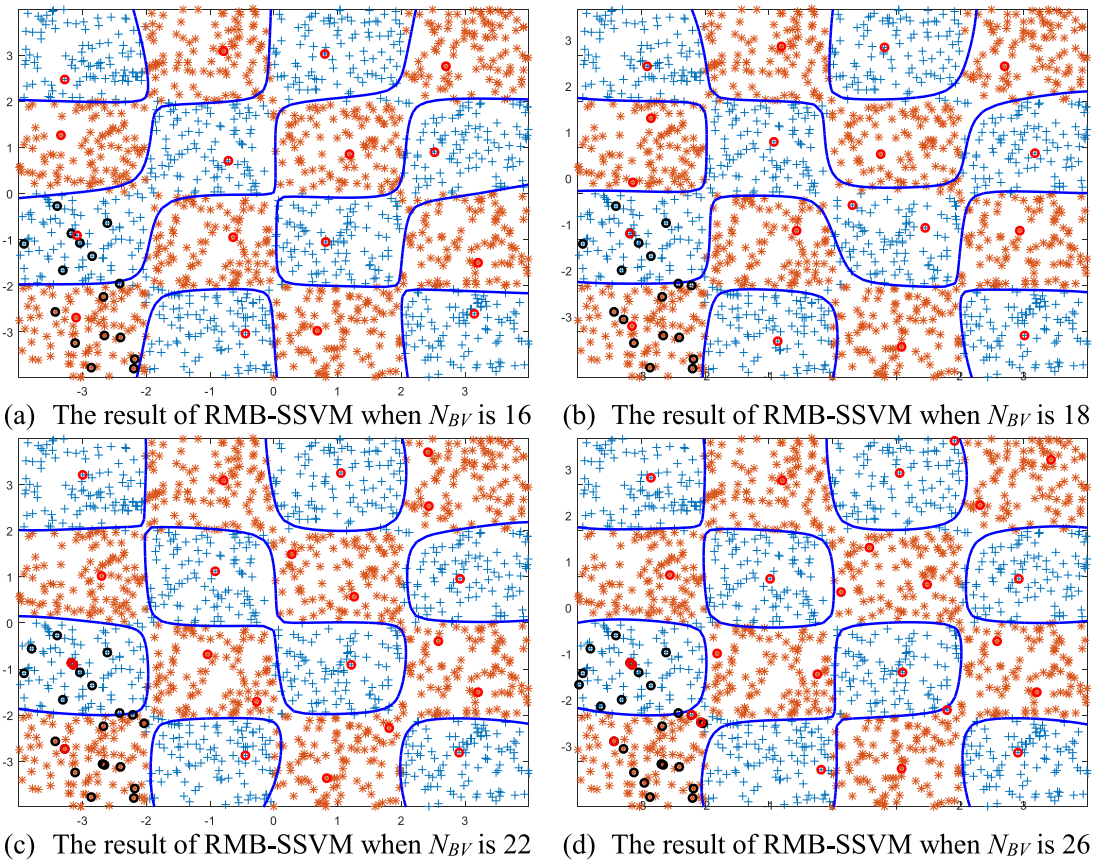


Fig. 3. The separating hyperplanes of RMB-SSVM under different numbers of the basis vectors.

Fig. 6 shows the classification accuracies under different regularization parameters when the parameter Δ is set to 1. We can find that the regularization parameter C severely influences the results of SVM. In other words, the accuracy is impaired for SVM when the parameter C is too large or small. As a contrast, the proposed methods, RMB-SSVM and MPD-SSVM, can give a good accuracy in a large range of the regularization parameter. Besides, the proposed methods, which use only 10 basis vectors, even give better results than SVM sometimes. The reason may be that some of the support vectors yielded by SVM are noise vectors, and so the quality of the decision function is contaminated.

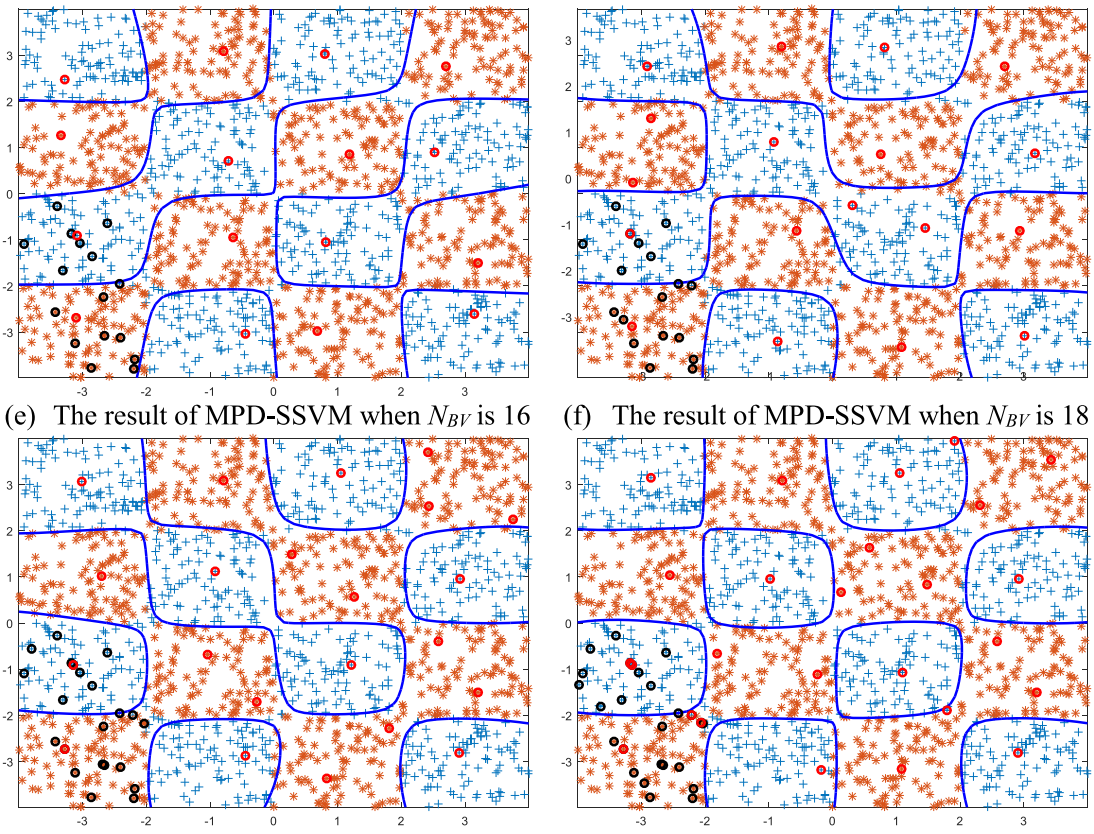
Further, we investigate the influence of the kernel parameter on the proposed methods. Here, we vary the kernel parameter σ in the set {0.05, 0.1, 0.5, 1, 2, 4, 8, 16}. Fig. 7(a) and (b) show the results on the Pima dataset under the regularization parameter C being 1 and 100. Obviously, when the kernel parameter σ is too large or small, the performance of the proposed methods would be impaired. The similar phenomena can be observed according to Fig. 7(c) and (d), which show the results on the German dataset.

Finally, we test the classification accuracies of the proposed methods vs the number of the used basis vectors in the case of fixing other parameters. Fig. 8(a) to (d) show the results on the Pima dataset when the regularization parameter C varies in the set {1, 10, 100, 1000}. Obviously, the proposed methods can give a relatively better accuracy when we use 10 basis vectors. This indicates the effectiveness of the proposed methods. Besides, we can observe that proposed methods can sometimes give better result than SVM although the used basis vectors are much less than the support vectors yielded by SVM, e.g. in Fig. 8(c) and (d). The reason, in our opinion, is that SVM would offer a poor result when the regularization parameter is not appropriate whereas the proposed methods can perform well in a large range of the regularization parameter according to Fig. 6. Fig. 8(e) to (f) show the results on the German dataset and we can observe the similar phenomena.

5.3. Experimental evaluations on benchmark datasets

In order to comprehensively evaluate the performances of the proposed methods, we conducted the experiments on ten UCI datasets [40]: Banana, Breast, Titanic, Waveform, German, Image, Diabetes, Ringnorm, Twonorm and Splice. The first six datasets have been used to test the performance of SLMC in [27,28]. Similarly to Section 5.2, we normalized each data by linearly scaling all features to the range [0, 1] before conducting experiments. In the experiments, we compared the proposed methods, i.e. RMB-SSVM and MPD-SSVM, with several related sparse methods including RSVM [24], SLMC [27,28] and SpSVM [26]. The key advantage of these methods is that they can control the sparsity level of the solution by setting the number of the used basis vectors in the decision. RSVM randomly selects some data points from the training set as the basis vectors. Here, we use it as a baseline. The other methods, including SLMC, SpSVM and the proposed methods, optimize the used basis vectors according to different criterions.

Following the manner in [27,28], we splits each dataset into the training and test subsets. Table 1 shows the characteristics of the used datasets in the experiments. For each split, we conduct the experiment by the following manner. First, we employ the 5-fold cross-validation technique to choose the parameter C from the set $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$ and the parameter σ from the set $\{2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{2}, 2^{3}\}$ for SVM on the training data. Then, we first conduct the training of SVM in term of the



(e) The result of MPD-SSVM when N_{BV} is 16 (f) The result of MPD-SSVM when N_{BV} is 18

(g) The result of MPD-SSVM when N_{BV} is 22 (h) The result of MPD-SSVM when N_{BV} is 26

Fig. 4. The separating hyperplanes of MPD-SSVM under different numbers of the basis vectors.

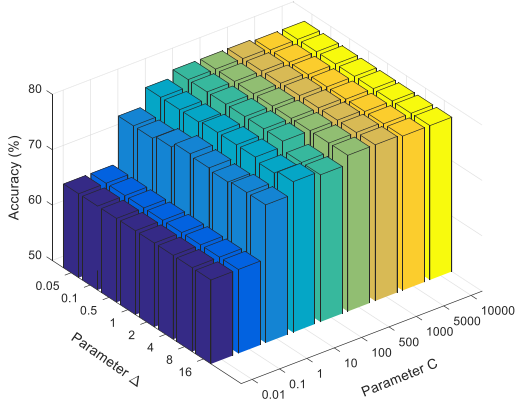
Table 1
The characteristic of the used datasets in the experiments.

Dataset	# of features	# of training data points	# of test data points	# of support vectors	Accuracy of SVM
Banana	2	400	4900	245.000+11.782	83.552+1.561
Breast	9	200	77	128.733+11.343	75.757+3.056
Titanic	3	150	2051	72.500+8.215	77.922+0.607
Waveform	21	400	4600	150.066+17.447	90.150+0.372
German	20	700	300	439.766+33.801	76.655+1.945
Image	18	1300	1010	169.233+19.531	97.507+0.520
Diabetis	8	468	300	262.733+21.771	78.733+1.660
Ringnorm	20	400	7000	92.200+19.387	97.869+0.188
Twonorm	20	400	7000	102.700+11.139	97.460+0.158
Splice	60	1000	2175	798.833+11.546	86.842+0.735

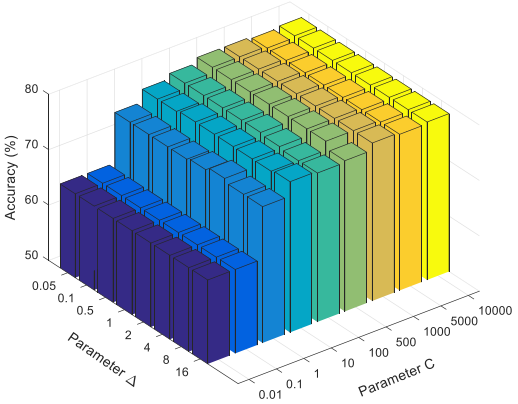
Table 2
Classification accuracy (%) comparison (*mean* ± *standard derivation*) when $N_{EV}/N_{SV} = 4\%$.

Dataset	RSVM	SLMC	SpSVM	RMB-SSVM	MPD-SSVM
Banana	64.155 ± 10.934	73.068 ± 6.578	71.697 ± 7.659	**77.761 ± 4.407**	77.574 ± 5.516
Breast	60.733 ± 10.644	69.310 ± 3.302	68.903 ± 3.170	72.059 ± 2.542	**72.571 ± 2.308**
Titanic	66.419 ± 5.840	71.486 ± 2.098	70.010 ± 4.655	**73.716 ± 3.142**	72.912 ± 3.852
Waveform	75.885 ± 6.573	82.729 ± 1.034	80.006 ± 3.102	**85.537 ± 2.435**	85.399 ± 5.497
German	64.853 ± 7.294	70.511 ± 3.244	68.512 ± 3.252	**73.028 ± 3.292**	72.873 ± 3.171
Image	71.845 ± 15.363	87.807 ± 2.568	85.934 ± 3.492	**90.188 ± 2.499**	88.774 ± 2.858
Diabetis	65.389 ± 10.076	69.826 ± 3.742	67.810 ± 4.846	73.176 ± 1.966	**73.313 ± 1.572**
Ringnorm	71.462 ± 15.443	90.905 ± 1.091	86.611 ± 2.690	**92.573 ± 2.510**	91.922 ± 2.537
Twonorm	73.960 ± 14.131	89.271 ± 1.557	87.590 ± 2.574	92.220 ± 2.160	**92.547 ± 1.804**
Splice	66.620 ± 12.957	80.850 ± 1.943	77.755 ± 3.135	**83.949 ± 1.732**	83.118 ± 1.678

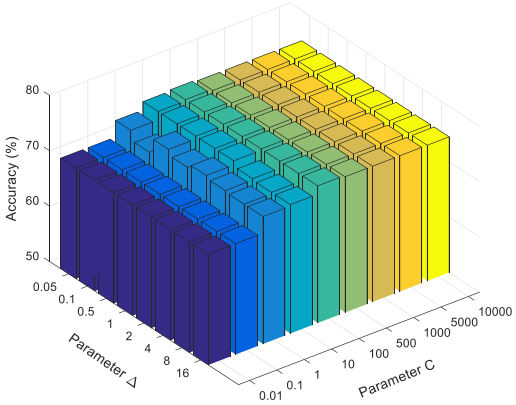
chosen parameter pair $\{C, \sigma\}$, and obtain the corresponding classification accuracy and the number (N_{SV}) of the support vectors according to the trained-well model. Table 1 shows the results of SVM on the selected datasets. Further, we run the several related competing methods, including RSVM, SLMC, SpSVM, RMB-SSVM and MPD-SSVM, under different sparsity levels (i.e. N_{EV}/N_{SV}). Here N_{SV} is the number of the support vectors obtained by SVM, and N_{EV} represents the number of the basis vectors in the several sparse methods. We test four sparsity levels: 4%, 6%, 8% and 10%. For each dataset, we repeat this process 50 times and compute the averaged classification accuracy and corresponding standard derivation.



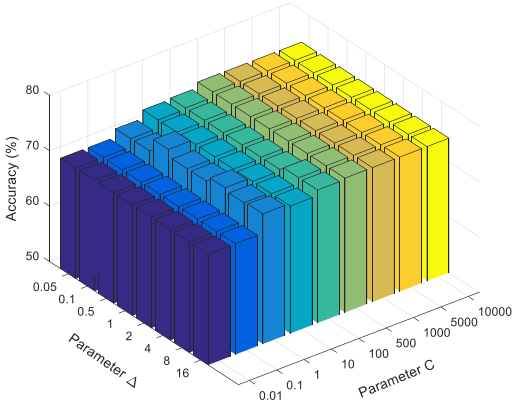
(a) Classification accuracies of RMB-SSVM on the Pima dataset



(b) Classification accuracies of MPD-SSVM on the Pima dataset

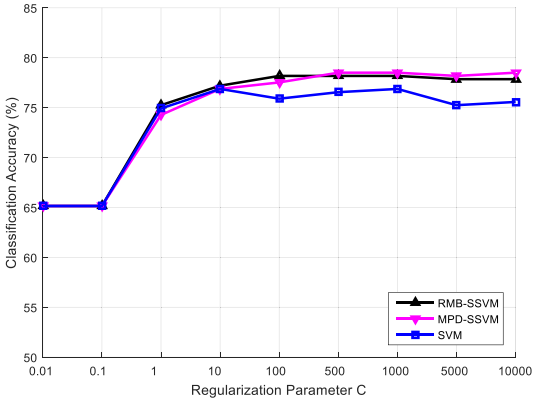


(c) Classification accuracies of RMB-SSVM on the German dataset

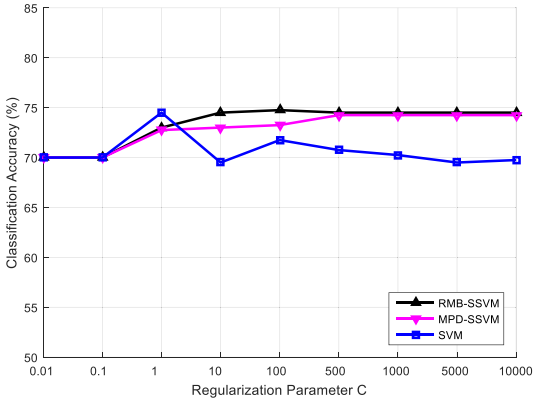


(d) Classification accuracies of MPD-SSVM on the German dataset

Fig. 5. Classification accuracies of the proposed methods under different Δ and C.



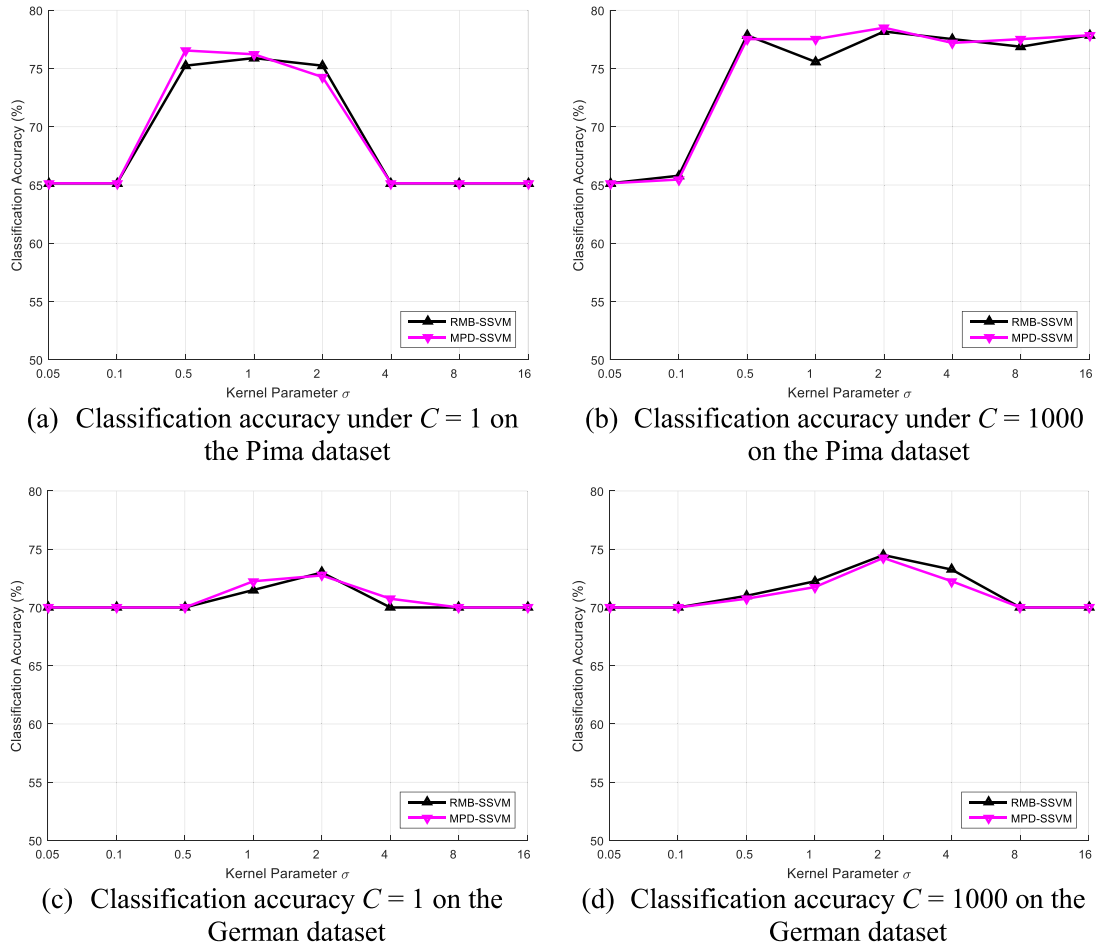
(a) Classification accuracies of the proposed methods on the Prima dataset



(b) Classification accuracies of the proposed methods on the German dataset

Fig. 6. Classification accuracies of the proposed methods under different regularization parameters and $\Delta = 1$.

Table 2 reports the experimental results of the several methods on the selected datasets under 4% sparsity levels. In contrast with the RSVM method, the other methods, including SLMC, SpSVM, RMB-SSVM and MPD-SSVM, improve distinctly the accuracies. This is because that the former randomly selects some data points from the training set as the basis vectors but the latter optimizes the finally used basis vectors according to some schemes. The proposed methods, RMB-SSVM and MPD-SSVM, further achieve better classification accuracies on the whole when comparing them with SLMC and SpSVM. The reason is that the conventional methods only embody the large margin principle, whereas our methods employ a criterion, which is related with the generalization performance of the large margin classifier, to guide the learning of the used basis vectors and corresponding coefficients. The similar phenomena can be observed in Tables 3–5, where the experimental results under 6%, 8% and 10% sparsity levels are shown. These experimental results indicate that the generalization performance can be improved by incorporating the



(a) Classification accuracy under $C = 1$ on the Pima dataset

(b) Classification accuracy under $C = 1000$ on the Pima dataset

(c) Classification accuracy $C = 1$ on the German dataset

(d) Classification accuracy $C = 1000$ on the German dataset

Fig. 7. Classification accuracies of the proposed methods under different kernel parameter.

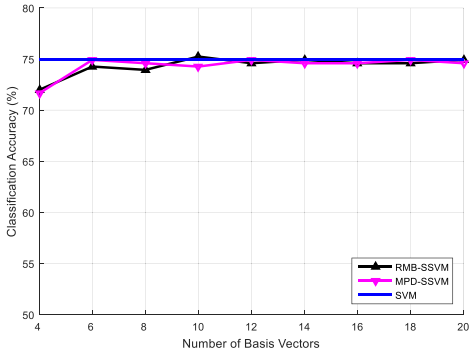
Table 3
Classification accuracy (%) comparison (*mean* ± *standard derivation*) when $N_{EV}/N_{SV} = 6\%$.

Dataset	RSVM	SLMC	SpSVM	RMB-SSVM	MPD-SSVM
Banana	68.325 ± 7.142	75.505 ± 4.712	75.873 ± 3.193	80.204 ± 0.758	**80.874 ± 2.621**
Breast	68.989 ± 5.379	72.966 ± 0.603	71.080 ± 1.516	**73.583 ± 1.539**	73.155 ± 1.942
Titanic	70.795 ± 3.338	74.330 ± 1.531	73.080 ± 1.339	**75.692 ± 1.308**	76.551 ± 2.112
Waveform	80.660 ± 3.215	83.546 ± 1.220	83.857 ± 1.818	**88.617 ± 0.791**	87.520 ± 0.885
German	68.033 ± 5.678	72.264 ± 2.427	70.033 ± 3.604	**75.713 ± 2.340**	74.987 ± 2.656
Image	79.525 ± 9.373	90.957 ± 1.537	89.457 ± 2.254	**94.345 ± 1.299**	93.976 ± 1.019
Diabetis	67.651 ± 5.591	72.282 ± 1.519	70.550 ± 1.122	**75.482 ± 1.688**	74.558 ± 2.629
Ringnorm	80.951 ± 9.608	93.567 ± 0.885	89.695 ± 1.531	**95.709 ± 2.693**	94.631 ± 1.540
Twonorm	82.144 ± 8.678	92.272 ± 0.922	90.678 ± 1.367	95.022 ± 2.119	**96.936 ± 1.892**
Splice	74.727 ± 6.357	82.273 ± 0.791	81.614 ± 0.731	**84.468 ± 1.371**	84.444 ± 1.481

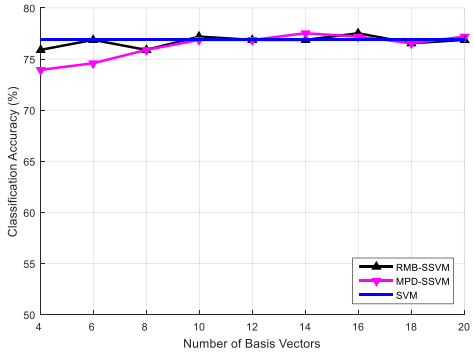
information of the radius of the minimum enclosing hypersphere when simplifying the solution of SVM. Our methods embodies the idea.

For a rigorous comparison, simultaneously, we further conducted the paired two-tailed t-test [41] on these methods when $N_{EV}/N_{SV} = 8\%$. Table 6 reports the p-value of t-test when comparing two paired methods. Here, we set a typical threshold of the p-value as 0.05. Obviously, RMB-SSVM has on the whole significant improvement in the generalized performance when being compared with RSVM, SLMC and SpSVM. However, its improvement in the generalized performance is not significant when comparing it with another our method MPD-SSVM. This means that MPD-SSVM is an effective method as substitution for RMB-SSVM. Besides, the classification accuracies of RMB-SSVM and SVM have no statistically significant difference. This means that the proposed methods can achieve a very close performance to one of SVM although the used basis vectors of RMB-SSVM are far less than the support vectors yielded by SVM.

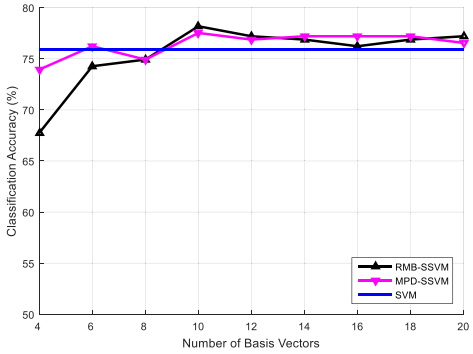
Finally, Table 7 reports the mean training time of several methods on the selected datasets when NEV/NSV = 10%. From Table 7, it is very clear that RSVM requires the least training time-cost in that it randomly selects a subset of the training data as the basis vectors and then only optimizes the corresponding coefficients. The other methods optimize the finally used basis vectors and require relatively more time to train the model. SLMC spends the most training time. The reason is that it uses a gradient descent scheme to update the basis vectors and so the optimization model is solved many times to search the



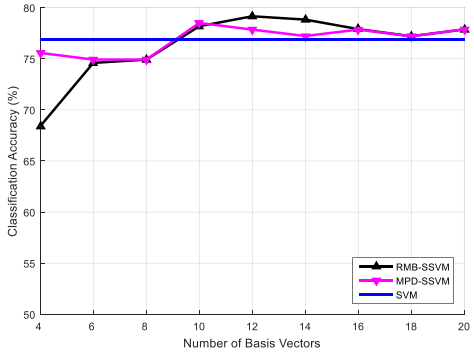
(a) Classification accuracy under $C = 1$ on the Pima dataset, NSV = 318



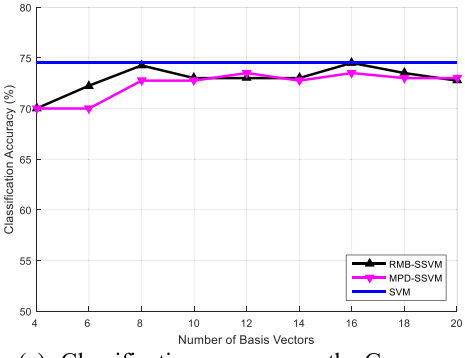
(b) Classification accuracy under $C = 10$ on the Pima dataset, NSV = 253



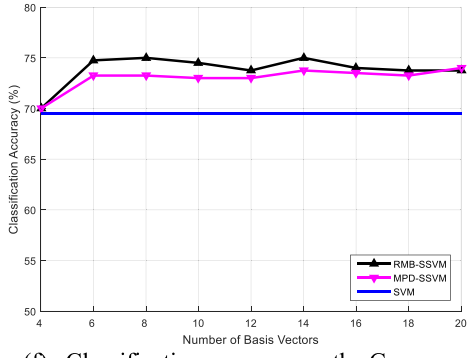
(c) Classification accuracy under $C = 100$ on the Pima dataset, NSV = 235



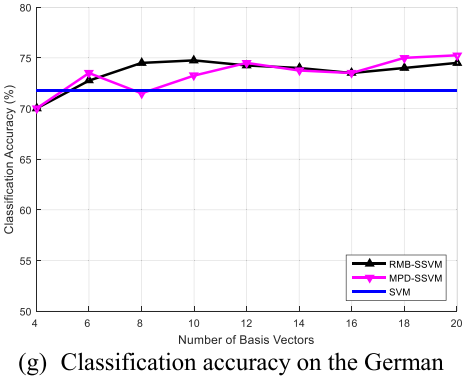
(d) Classification accuracy under $C = 1000$ on the Pima dataset, NSV = 224



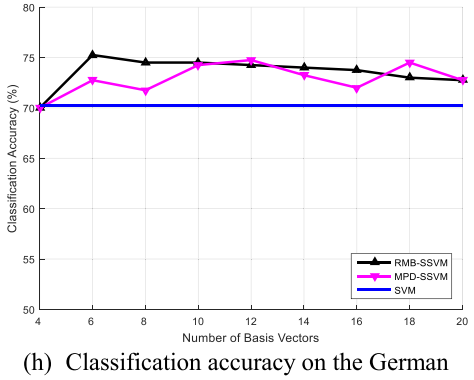
(e) Classification accuracy on the German dataset under $C = 1$,NSV = 368



(f) Classification accuracy on the German dataset under $C = 10$,NSV = 334



(g) Classification accuracy on the German dataset under $C = 100$,NSV = 319



(h) Classification accuracy on the German dataset under $C = 1000$,NSV = 309

Fig. 8. Classification accuracies of the proposed methods under different numbers of the basis vector.

Table 4
Classification accuracy (%) comparison (*mean* ± *standard derivation*) when $N_{EV}/N_{SV} = 8\%$.

Dataset	RSVM	SLMC	SpSVM	RMB-SSVM	MPD-SSVM
Banana	75.184 ± 3.042	79.435 ± 1.723	76.691 ± 2.482	**83.476 ± 0.880**	83.180 ± 0.957
Breast	70.143 ± 2.735	73.014 ± 0.526	73.129 ± 0.508	**75.727 ± 0.436**	75.044 ± 0.240
Titanic	71.817 ± 2.322	75.113 ± 1.204	75.130 ± 0.974	77.692 ± 0.377	**77.709 ± 0.678**
Waveform	81.592 ± 3.121	85.595 ± 1.942	86.087 ± 1.498	90.125 ± 0.576	**90.228 ± 0.543**
German	71.213 ± 3.029	73.042 ± 2.009	72.689 ± 2.647	**76.502 ± 0.407**	76.437 ± 0.444
Image	85.827 ± 4.133	93.387 ± 1.424	94.304 ± 1.340	**97.561 ± 0.761**	97.266 ± 0.805
Diabetis	70.084 ± 2.556	75.562 ± 1.988	74.704 ± 2.544	**78.950 ± 1.890**	78.024 ± 1.753
Ringnorm	85.191 ± 4.959	94.515 ± 0.775	92.059 ± 1.371	**97.618 ± 0.718**	97.387 ± 0.667
Twonorm	86.936 ± 5.018	94.473 ± 1.063	91.875 ± 2.002	97.401 ± 0.951	**97.406 ± 0.521**
Splice	78.069 ± 5.310	83.002 ± 0.844	83.149 ± 0.929	**86.798 ± 0.679**	86.276 ± 0.867

Table 5
Classification accuracy (%) comparison (*mean* ± *standard derivation*) when $N_{EV}/N_{SV} = 10\%$.

Dataset	RSVM	SLMC	SpSVM	RMB-SSVM	MPD-SSVM
Banana	78.210 ± 1.994	80.038 ± 1.686	79.623 ± 0.918	**83.630 ± 0.859**	83.288 ± 0.758
Breast	72.762 ± 2.170	73.933 ± 0.611	73.533 ± 0.589	75.465 ± 0.211	**75.622 ± 0.217**
Titanic	72.119 ± 2.434	76.004 ± 1.530	75.146 ± 1.354	**77.876 ± 0.624**	77.852 ± 0.734
Waveform	85.085 ± 2.130	87.425 ± 1.727	88.510 ± 1.430	**90.092 ± 0.843**	90.072 ± 0.727
German	73.322 ± 3.308	75.199 ± 0.552	75.169 ± 0.864	76.435 ± 0.609	**76.594 ± 0.711**
Image	87.796 ± 2.954	94.081 ± 1.431	95.804 ± 1.470	**97.551 ± 0.701**	97.188 ± 1.073
Diabetis	72.742 ± 3.214	75.960 ± 2.961	76.554 ± 1.613	**78.525 ± 1.224**	78.383 ± 1.017
Ringnorm	89.644 ± 2.402	95.188 ± 1.569	96.965 ± 1.289	**97.771 ± 0.847**	97.360 ± 0.784
Twonorm	89.094 ± 4.612	94.877 ± 0.930	96.493 ± 0.793	**97.448 ± 0.713**	97.380 ± 0.428
Splice	81.089 ± 3.924	84.714 ± 0.412	85.312 ± 0.578	**86.476 ± 0.357**	86.360 ± 0.474

Table 6
P-value of *t*-test when $N_{EV}/N_{SV} = 8\%$.

Dataset	RMB-SSVM/RSVM	RMB-SSVM/SLMC	RMB-SSVM/SpSVM	RMB-SSVM/MPD-SSVM	RMB-SSVM/SVM
Banana	**0.00829**	**0.00947**	**0.03378**	0.07339	0.15590
Breast	**0.00904**	**0.01821**	**0.01634**	0.09309	0.08541
Titanic	**0.02530**	**0.02057**	**0.03604**	0.06391	0.23479
Waveform	**0.01739**	**0.03420**	**0.04344**	0.13013	0.14460
German	**0.02703**	**0.01665**	**0.02844**	0.06955	0.09542
Image	**0.02085**	**0.04210**	**0.03589**	0.23329	0.08771
Diabetis	**0.01650**	**0.03737**	**0.02856**	**0.04671**	0.06628
Ringnorm	**0.02403**	**0.03521**	**0.04134**	0.16482	0.23308
Twonorm	**0.01170**	**0.03312**	**0.04776**	0.09252	0.08985
Splice	**0.02060**	**0.03287**	**0.03986**	0.08422	0.11182

suitable step. In contrast with SpSVM, our methods, MPDSLMC and RMBSLMC, require more training time but can give better classification accuracy according to Tables 2–6. This is because our methods further incorporates the radius information of the minimum hypersphere enclosing the data.

5.4. Experimental evaluations on large-scale datasets

In this subsection, we will further compare the proposed methods with the other methods on several large-scale datasets. In the experiments, the used datasets includes Adult, IJCNN, Shuttle and Vehicle, and they are available from the following LIBSVM-Tools page: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. For the Adult dataset, which contains 32,561 examples with 123 binary features, we randomly chosen 20,000 data points for training and the rest for test. The IJCNN dataset has 141,691 examples and each example is described by 22 features. In the experiment on this dataset, we randomly chosen 50,000 examples as the training data. Shuttle consists of 58,000 examples with 9 features. In this dataset, 40,000 examples are randomly chosen for training. This is a multiclass dataset with seven classes. Here, we solved only the binary classification problem of differentiating class 1 from the rest. As for the Vehicle dataset, which has 98,528 examples with 100 features, we randomly chosen 50,000 examples for training and the rest for test. This dataset consists of three classes. Here, we deal only with the binary classification task of differentiating class 1 from the other.

In the experiments, we first carry out the SVM method on the selected datasets, and compute the classification accuracies and the numbers of the yielded support vectors. Table 8 shows the characteristics of the used datasets in the experiments and the experimental results of SVM. Here, we set C = 10 and $\sigma = 2$ for Adult and Vehicle, and C = 10 and $\sigma = 0.5$ for IJCNN and Shuttle.

Further, we compare the proposed methods with the other several methods under different numbers of the used basis vectors. Here, we vary the number of the used basis vectors in the set {5, 10, 15, 20, 25, 30, 35, 40, 45, 50}. For each method, we repeat the training-test process 10 times and compute the averaged classification accuracy. Fig. 9 shows the averaged classification accuracies of several methods vs the number of the used basis vectors. Obviously, the accuracies of several methods are improved with the used basis vectors increase. Our methods, RMBRSVM and MPDRSVM, achieve a significant improvement in the accuracy in contrast with the other methods. The reason is that the proposed method employ a learning criterion, which integratedly considers the information of the margin of the separating hyperplane and the radius of the minimum enclosing hypersphere. Besides, RMBRSVM and MPDRSVM show the close generalization performance on the whole. This demonstrates that it is effective to approximate the radius by using the maximum pairwise distance over the pairs of data points in our methodology.

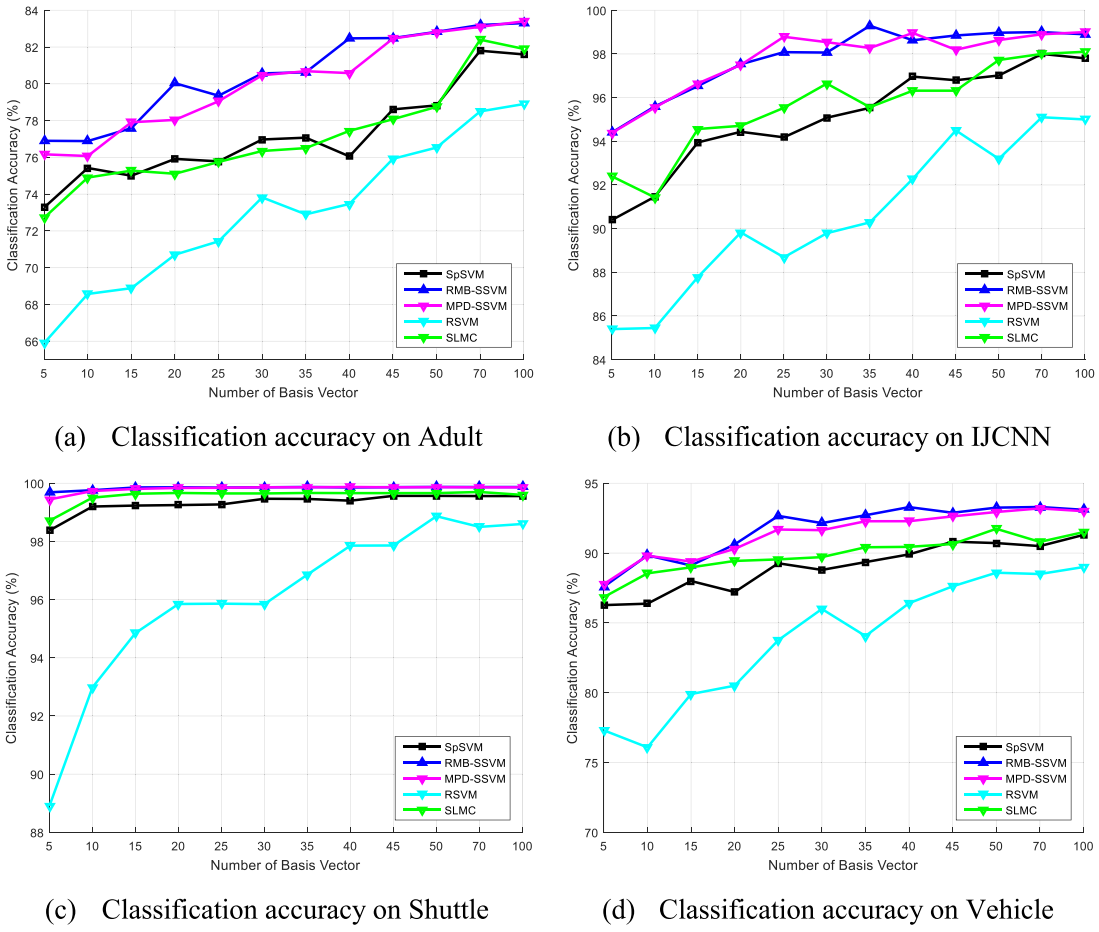


Fig. 9. Classification accuracies of the proposed methods under different numbers of basis vector on the selected datasets.

Table 7
Training time comparison on the selected datasets (in seconds).

Dataset	RSVM	SLMC	SpSVM	RMB-SSVM	MPD-SSVM
Banana	**0.483**	39.504	5.891	44.342	29.182
Breast	**0.242**	12.636	2.813	11.298	9.694
Titanic	**0.221**	10.329	3.319	11.235	6.865
Waveform	**2.838**	231.646	1.034	269.492	117.896
German	**2.659**	769.353	2.166	694.168	397.627
Image	**9.927**	1884.328	85.114	1202.011	798.449
Diabetis	**3.045**	625.684	24.447	415.355	345.995
Ringnorm	**2.106**	541.757	25.975	462.163	328.698
Twonorm	**2.104**	509.779	26.323	474.871	327.659
Splice	**13.389**	2418.533	128.875	1813.248	1433.714

Table 8
The characteristics of the used datasets and the experimental result of SVM.

Dataset	# of features	# of training data points	# of test data points	# of support vectors	Accuracy of SVM
Adult	123	20,000	12,561	8098	82.8596
IJCNN	22	50,000	91,691	2965	99.206
Shuttle	9	40,000	18,000	1012	99.8944
Vehicle	100	50,000	48,528	9297	93.6243

6. Conclusions

In this paper, we proposed a novel method named as RMB-SSVM to condense the basis vectors used in the testing phase of SVM. The key feature of RMB-SSVM is that it exploits integratedly the information of the margin of the separating hyperplane and the radius of the minimum enclosing hypersphere to guide the selecting of the basis vectors and the learning of the corresponding coefficients in the decision function. Further, we explored how to approximate the radius in our methodology and developed the MPD-SSVM method. The comprehensive experimental results well demonstrated the effectiveness of the proposed methods.

6.1. Limitations

The main limitations of our work displays in following two aspects: (1) we employ an alternate optimization strategy to handle the proposed optimization models. This cannot guarantee that the procedure can offer the global optimal solution and the initial values might impact the solution quality. (2) Similarly to other kernel methods, we need to empirically manually set the related parameters when applying the proposed methods. Thus, the inappropriate parameters would limit the generalization performances of the proposed methods.

6.2. Future directions

According to the main limitations of the proposed methods, one of the future works is to further explore the more effective way to deal with the proposed optimization models. Besides, the proposed methods directly exploit the radius-margin bound, which characterizes the generalization ability of the classifier and is generally used to guide the selecting of the hyperparameters of SVM. Thus, it is worth to study whether the proposed methods can be extended to simultaneously optimize the regularization and kernel parameters during learning the basis vectors

and the corresponding coefficients. This will further benefit the application of the proposed methods.

CRediT authorship contribution statement

Xiaoming Wang: Methodology, Conceptualization, Software, Writing - original draft. **Shitong Wang:** Writing - review & editing. **Zengxi Huang:** Visualization. **Yajun Du:** Funding acquisition, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported in part by the National Science Foundation of China (Grant No. 61532009).

Appendix. Computation of the inner product $\left(\partial k_{\mathcal{Z}}\left(\mathbf{x}_i, \mathbf{x}_j\right)/\partial\varphi\left(\mathbf{z}_u\right)\right)^T \varphi\left(\mathbf{x}_k\right)$

For convenience, we set $\hat{\mathbf{z}}_i = \varphi\left(\mathbf{z}_i\right)$ and $\hat{\mathbf{x}}_i = \varphi\left(\mathbf{x}_i\right)$. Thus, we can reformulate respectively $\varphi_{\mathcal{Z}}\left(\mathbf{x}_i\right)$ and $\mathbf{K}_{\mathcal{Z}}$ as

$$\begin{aligned}\varphi_{\mathcal{Z}}\left(\mathbf{x}_i\right) &= \left[\varphi^T\left(\mathbf{x}_i\right)\varphi\left(\mathbf{z}_1\right), \ldots, \phi^T\left(\mathbf{x}_i\right)\varphi\left(\mathbf{z}_{N_{BV}}\right)\right]^T \\ &= \left[\hat{\mathbf{x}}_i^T\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{x}}_i^T\hat{\mathbf{z}}_{N_{BV}}\right]^T\end{aligned} \tag{40}$$

and

$$\mathbf{K}_{\mathcal{Z}} = \left(\varphi_{\mathcal{Z}}^T\left(\mathbf{z}_i\right)\varphi_{\mathcal{Z}}\left(\mathbf{z}_j\right)\right)_{N_{BV}\times N_{BV}} = \left(\hat{\mathbf{z}}_i^T\hat{\mathbf{z}}_j\right)_{N_{BV}\times N_{BV}} \tag{41}$$

Further, we can express the derivative of $\varphi_Z\left(\mathbf{x}_i\right)$ with respect to $\hat{z}_{u,v}$ as follows

$$\begin{aligned}\frac{\partial\varphi_{\mathcal{Z}}\left(\mathbf{x}_i\right)}{\partial\hat{z}_{u,v}} &= \left[\frac{\partial\left(\hat{\mathbf{x}}_i^T\hat{\mathbf{z}}_1\right)}{\partial\hat{z}_{u,v}}, \ldots, \frac{\partial\left(\hat{\mathbf{x}}_i^T\hat{\mathbf{z}}_u\right)}{\partial\hat{z}_{u,v}}, \ldots, \frac{\partial\left(\hat{\mathbf{x}}_i^T\hat{\mathbf{z}}_{N_{BV}}\right)}{\partial\hat{z}_{u,v}}\right]^T \\ &= \left[0, \ldots, \hat{x}_{i,v}, \ldots, 0\right]^T\end{aligned} \tag{42}$$

Here $\hat{z}_{u,v}$ and $\hat{x}_{i,v}$ represent the vth component of the vectors $\hat{\mathbf{z}}_u$ and $\hat{\mathbf{x}}_i$, respectively. Similarly, we have

$$\begin{aligned}\frac{\partial\mathbf{K}_{\mathcal{Z}}}{\partial\hat{z}_{u,v}} &= \left(\frac{\partial\left(\hat{\mathbf{z}}_i^T\hat{\mathbf{z}}_j\right)}{\partial\hat{z}_{u,v}}\right)_{N_{BV}\times N_{BV}} \\ &= \begin{bmatrix} \mathbf{0} & \frac{\partial\left(\hat{\mathbf{z}}_1^T\hat{\mathbf{z}}_u\right)}{\partial\hat{z}_{u,v}} & \mathbf{0} \\ \frac{\partial\left(\hat{\mathbf{z}}_u^T\hat{\mathbf{z}}_1\right)}{\partial\hat{z}_{u,v}} & \frac{\partial\left(\hat{\mathbf{z}}_u^T\hat{\mathbf{z}}_u\right)}{\partial\hat{z}_{u,v}} & \frac{\partial\left(\hat{\mathbf{z}}_u^T\hat{\mathbf{z}}_{N_{BV}}\right)}{\partial\hat{z}_{u,v}} \\ \mathbf{0} & \frac{\partial\left(\hat{\mathbf{z}}_{N_{BV}}^T\hat{\mathbf{z}}_u\right)}{\partial\hat{z}_{u,v}} & \mathbf{0} \end{bmatrix}\end{aligned} \tag{43}$$

where $\mathbf{0}$ represents a matrix with proper size and

$$\frac{\partial\left(\hat{\mathbf{z}}_i^T\hat{\mathbf{z}}_j\right)}{\partial\hat{z}_{u,v}} = \begin{cases}\hat{z}_{j,v}, & i = u\ and\ j \neq u \\ \hat{z}_{i,v}, & i \neq u\ and\ j = u \\ 2\hat{z}_{u,v}, & i = u\ and\ j = u \\ 0, & otherwise\end{cases} \tag{44}$$

Thus, according to (42) and (44), we have

$$\begin{aligned}&\frac{\partial k_{\mathcal{Z}}\left(\mathbf{x}_i, \mathbf{x}_j\right)}{\partial\hat{z}_{u,v}} \\ &= \frac{\partial\left(\varphi_{\mathcal{Z}}^T\left(\mathbf{x}_i\right)\mathrm{K}_{\mathcal{Z}}^{-1}\varphi_{\mathcal{Z}}\left(\mathbf{x}_j\right)\right)}{\partial\hat{z}_{u,v}} \\ &= \frac{\partial\varphi_{\mathcal{Z}}^T\left(\mathbf{x}_i\right)}{\partial\hat{z}_{u,v}}\mathrm{K}_{\mathcal{Z}}^{-1}\varphi_{\mathcal{Z}}\left(\mathbf{x}_j\right) + \varphi_Z^T\left(\mathbf{x}_i\right)\frac{\partial\mathrm{K}_{\mathcal{Z}}^{-1}}{\partial\hat{z}_{u,v}}\varphi_Z\left(\mathbf{x}_j\right) \\ &\quad + \varphi_Z^T\left(\mathbf{x}_i\right)\mathrm{K}_{\mathcal{Z}}^{-1}\frac{\partial\varphi_{\mathcal{Z}}\left(\mathbf{x}_j\right)}{\partial\hat{z}_{u,v}} \\ &= \frac{\partial\varphi_{\mathcal{Z}}^T\left(\mathbf{x}_i\right)}{\partial\hat{z}_{u,v}}\boldsymbol{\lambda}_j - \varphi_Z^T\left(\mathbf{x}_i\right)\mathrm{K}_{\mathcal{Z}}^{-1}\frac{\partial\mathrm{K}_Z}{\partial\hat{z}_{u,v}}\mathrm{K}_{\mathcal{Z}}^{-1}\varphi_{\mathcal{Z}}\left(\mathbf{x}_j\right) + \boldsymbol{\lambda}_i^T\frac{\partial\varphi_{\mathcal{Z}}\left(\mathbf{x}_j\right)}{\partial\hat{z}_{u,v}} \\ &= \frac{\partial\varphi_{\mathcal{Z}}^T\left(\mathbf{x}_i\right)}{\partial\hat{z}_{u,v}}\boldsymbol{\lambda}_j - \boldsymbol{\lambda}_i^T\frac{\partial\mathrm{K}_{\mathcal{Z}}}{\partial\hat{z}_{u,v}}\boldsymbol{\lambda}_j + \boldsymbol{\lambda}_i^T\frac{\partial\varphi_{\mathcal{Z}}\left(\mathbf{x}_j\right)}{\partial\hat{z}_{u,v}} \\ &= \lambda_{j,u}\hat{x}_{i,v} - \sum_{t=1}^{N_{BV}}\left(\lambda_{i,u}\lambda_{j,t} + \lambda_{j,u}\lambda_{i,t}\right)\hat{z}_{t,v} + \lambda_{i,u}\hat{x}_{j,v}\end{aligned} \tag{45}$$

Here $\boldsymbol{\lambda}_i = \mathrm{K}_{\mathcal{Z}}^{-1}\varphi_{\mathcal{Z}}\left(\mathbf{x}_i\right)$ and $\lambda_{i,u}$ refers to the uth component of the vector $\boldsymbol{\lambda}_i$.

According to (45) and $\hat{\mathbf{z}}_u = \varphi\left(\mathbf{z}_u\right)$, we can formulate $\partial k_{\mathcal{Z}}\left(\mathbf{x}_i, \mathbf{x}_j\right)/\partial\varphi\left(\mathbf{z}_u\right)$ as follows

$$\begin{aligned}&\frac{\partial k_{\mathcal{Z}}\left(\mathbf{x}_i, \mathbf{x}_j\right)}{\partial\varphi\left(\mathbf{z}_u\right)} \\ &= \lambda_{j,u}\hat{\mathbf{x}}_i - \sum_{t=1}^{N_{BV}}\left(\lambda_{i,u}\lambda_{j,t} + \lambda_{j,u}\lambda_{i,t}\right)\hat{\mathbf{z}}_t + \lambda_{i,u}\hat{\mathbf{x}}_j \\ &= \lambda_{j,u}\varphi\left(\mathbf{x}_i\right) - \sum_{t=1}^{N_{BV}}\left(\lambda_{i,u}\lambda_{j,t} + \lambda_{j,u}\lambda_{i,t}\right)\varphi\left(\mathbf{z}_t\right) + \lambda_{i,u}\varphi\left(\mathbf{x}_j\right)\end{aligned} \tag{46}$$

Finally, the inner product between $\partial k_{\mathcal{Z}}\left(\mathbf{x}_i, \mathbf{x}_j\right)/\partial\varphi\left(\mathbf{z}_u\right)$ and a vector $\varphi\left(\mathbf{x}_k\right)$ in the feature space can be formulated as

$$\begin{aligned}&\left(\frac{\partial k_{\mathcal{Z}}\left(\mathbf{x}_i, \mathbf{x}_j\right)}{\partial\varphi\left(\mathbf{z}_u\right)}\right)^T\varphi\left(\mathbf{x}_k\right) \\ &= \left(\lambda_{j,u}\varphi\left(\mathbf{x}_i\right) - \sum_{t=1}^{N_{BV}}\left(\lambda_{i,u}\lambda_{j,t} + \lambda_{j,u}\lambda_{i,t}\right)\varphi\left(\mathbf{z}_t\right) + \lambda_{i,u}\varphi\left(\mathbf{x}_j\right)\right)^T\varphi\left(\mathbf{x}_k\right) \\ &= \lambda_{j,u}\varphi^T\left(\mathbf{x}_i\right)\varphi\left(\mathbf{x}_k\right) - \sum_{t=1}^{N_{BV}}\left(\lambda_{i,u}\lambda_{j,t} + \lambda_{j,u}\lambda_{i,t}\right)\varphi^T\left(\mathbf{z}_t\right)\varphi\left(\mathbf{x}_k\right) \\ &\quad + \lambda_{i,u}\varphi^T\left(\mathbf{x}_j\right)\varphi\left(\mathbf{x}_k\right) \\ &= \lambda_{j,u}k\left(\mathbf{x}_i, \mathbf{x}_k\right) + \lambda_{i,u}k\left(\mathbf{x}_j, \mathbf{x}_k\right) - \sum_{t=1}^{N_{BV}}\left(\lambda_{i,u}\lambda_{j,t} + \lambda_{j,u}\lambda_{i,t}\right)k\left(\mathbf{z}_t, \mathbf{x}_k\right)\end{aligned} \tag{47}$$

References

[1] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[2] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discov. 2 (2) (1998) 121–167.
[3] M. Jändel, Biologically relevant neural network architectures for support vector machines, Neural Netw. 49 (2014) 39–50.
[4] C.R. Huang, Y.T. Chen, W.Y. Chen, H.C. Cheng, B.S. Sheu, Gastroesophageal reflux disease diagnosis using hierarchical heterogeneous descriptor fusion support vector machine, IEEE Trans. Biomed. Eng. 63 (3) (2016) 588–599.
[5] X. Li, X. Jia, L. Wang, K. Zhao, On spectral unmixing resolution using extended support vector machines, IEEE Trans. Geosci. Remote Sens. 53 (9) (2016) 4985–4996.
[6] A.H.A. El-Atta, A.E. Hassanien, Two-class support vector machine with new kernel function based on paths of features for predicting chemical activity, Inform. Sci. 403-404 (2017) 42–54.
[7] J. Tang, D. Li, Y. Tian, D. Liu, Multi-view learning based on nonparallel support vector machine, Knowl.-Based Syst. 158 (2018) 94–108.

[8] J. Zhao, Y. Xu, H. Fujita, An improved non-parallel Universum support vector machine and its safe sample screening rule, Knowl.-Based Syst. 170 (2019) 79–88.
[9] T. Zhou, H. Lu, W. Wang, X. Yong, GA-SVM based feature selection and parameter optimization in hospitalization expense modeling, Appl. Soft Comput. 75 (2019) 323–332.
[10] V. Franc, V. Hlaváč, An iterative algorithm learning the maximal margin classifier, Pattern Recognit. 36 (2003) 1985–1996.
[11] R.E. Fan, P.H. Chen, C.J. Lin, Working set selection using second order information for training support vector machines, J. Mach. Learn. Res. 6 (2005) 1889–1918.
[12] I.W. Tsang, J.T. Kwok, P.M. Cheung, Core vector machines: fast SVM training on very large data sets, J. Mach. Learn. Res. 6 (2005) 363–392.
[13] M. Varewyck, J.P. Martens, A practical approach to model selection for support vector machines with a gaussian kernel, IEEE Trans. Syst. Man Cybern. B 41 (2) (2011) 330–340.
[14] J. López, A. Barbero, J.R. Dorronsoro, Clipping algorithms for solving the nearest point problem over reduced convex hulls, Pattern Recognit. 44 (2011) 607–614.
[15] S. Avidan, Support vector tracking, IEEE Trans. Pattern Anal. Mach. Intell. 26 (8) (2004) 1064–1072.
[16] Y. Liu, Y.F. Zheng, Soft SVM and its application in video-object extraction, IEEE Trans. Signal Process. 55 (7) (2007) 3272–3282.
[17] M. Tan, G. Pan, Y. Wang, Y. Zhang, Z. Wu, L1-norm latent SVM for compact features in object detection, Neurocomputing 139 (2) (2014) 56–64.
[18] S. Zhang, Y. Sui, X. Yu, S. Zhao, L. Zhang, Hybrid support vector machines for robust object tracking, Pattern Recognit. 48 (8) (2015) 2474–2488.
[19] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M. Cheng, S.L. Hicks, P.H.S. Torr, Struck: structured output tracking with kernels, IEEE Trans. Pattern Anal. Mach. Intell. 38 (10) (2016) 2096–2109.
[20] S. Zhang, Y. Sui, S. Zhao, L. Zhang, Graph-regularized structured support vector machine for object tracking, IEEE Trans. Circuits Syst. Video Technol. 27 (6) (2017) 1249–1262.
[21] T. Downs, K.E. Gates, A. Masters, Exact simplification of support vector solutions, J. Mach. Learn. Res. 2 (2001) 2293–2297.
[22] D. Geebelen, J.A.K. Suykens, J. Vandewalle, Reducing the number of support vectors of SVM classifiers using the smoothed separable case approximation, IEEE Trans. Neural Netw. Learn. Syst. 23 (4) (2012) 682–688.
[23] C.J.C. Burges, Simplified support vector decision rules, in: Proceedings of the 13th International Conference on Machine Learning, 1996, pp. 71-77.
[24] Y.J. Lee, O.L. Mangasarian, RSVM: reduced support vector machines, in: Proceedings of the First SIAM International Conference on Data Mining, 2001, pp. 1-17.
[25] K.M. Lin, C.J. Lin, A study on reduced support vector machines, IEEE Trans. Neural Netw. 14 (2003) 1449–1459.
[26] S.S. Keerthi, O. Chapelle, D. DeCoste, Building support vector machines with reduced classifier complexity, J. Mach. Learn. Res. 7 (2006) 1493–1515.
[27] M. Wu, B. Schölkopf, G. Bakir, Building sparse large margin classifiers, in: Proceedings of the 22th International Conference on Machine Learning, 2005, pp. 996-1003.
[28] M. Wu, B. Schölkopf, G. Bakir, A direct method for building sparse kernel learning algorithms, J. Mach. Learn. Res. 7 (2006) 603–624.
[29] M. Hu, Y. Chen, J.T.Y. Kwok, Building sparse multiple-kernel SVM classifiers, IEEE Trans. Neural Netw. 20 (5) (2009) 827–839.
[30] V. Vapnik, O. Chapelle, Bounds on error expectation for support vector machines, Neural Comput. 12 (9) (2000) 2013–2036.
[31] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Mach. Learn. 46 (1–3) (2002) 131–159.
[32] S.S. Keerthi, Efficient tuning of SVM hyperparameters using radius-margin bound and iterative algorithms, IEEE Trans. Neural Netw. 13 (5) (2002) 1225–1229.
[33] K. Duan, S.S. Keerthi, A.N. Poo, Evaluation of simple performance measures for tuning SVM hyperparameters, Neurocomputing 51 (2003) 41–59.
[34] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, 2004.
[35] K.M. Chung, W.C. Kao, C.L. Sun, L.L. Wang, C.J. Lin, Radius margin bounds for support vector machines with the RBF kernel, Neural Comput. 15 (11) (2003) 2643–2681.
[36] D.M.J. Tax, R.P.W. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[37] P. Sun, X. Yao, Sparse approximation through boosting for learning large scale kernel machines, IEEE Trans. Neural Netw. 21 (6) (2010) 883–894.
[38] H. Do, A. Kalousis, Convex formulations of radius-margin based support vector machines, in: Proceedings of the 30th International Conference on Machine Learning (ICML), 2013, pp. 169-177.
[39] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, in: ACM Transactions on Intelligent Systems and Technology, 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[40] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2019, [http://archive.ics.uci.edu/ml].
[41] E. Alpaydin, Introduction to Machine Learning, The MIT Press, Cambridge, MA, USA, 2004.