Soft Computing (2022) 26:6591–6618
https://doi.org/10.1007/s00500-022-07040-9

OPTIMIZATION

Ramp loss KNN-weighted multi-class twin support vector machine

Huiru Wang[1] · Yitian Xu[2] · Zhijian Zhou[2]

Accepted: 10 March 2022 / Published online: 6 May 2022

Abstract
The K-nearest neighbor-weighted multi-class twin support vector machine (KWMTSVM) is an effective multi-classification algorithm which utilizes the local information of all training samples. However, it is easily affected by the noises and outliers owing to the use of the hinge loss function. That is because the outlier will obtain a huge loss and become the support vector, which will shift the separating hyperplane inappropriately. To reduce the negative influence of outliers, we use the ramp loss function to replace the hinge loss function in KWMTSVM and propose a novel sparse and robust multi-classification algorithm named ramp loss K-nearest neighbor-weighted multi-class twin support vector machine (RKWMTSVM) in this paper. Firstly, the proposed RKWMTSVM restricts the loss of outlier to a fixed value, thus the negative influence on the construction of hyperplane is suppressed and the classification performance is further improved. Secondly, since outliers will not become support vectors, the RKWMTSVM is a sparser algorithm, especially compared with KWMTSVM. Thirdly, because RKWMTSVM is a non-differentiable non-convex optimization problem, we use the concave–convex procedure (CCCP) to solve it. In each iteration of CCCP, the proposed RKWMTSVM solves a series of KWMTSVM-like problems. That also means RKWMTSVM inherits the merits of KWMTSVM, namely, it can exploit the local information of intra-class to improve the generalization ability and use inter-class information to remove the redundant constraints and accelerate the solution process. In the end, the clipping dual coordinate descent (clipDCD) algorithm is employed into our RKWMTSVM to further speed up the computational speed. We do numerical experiments on twenty-four benchmark datasets. The experimental results verify the validity and effectiveness of our algorithm.

Keywords Weighted-TSVM · Hinge loss · Ramp loss · Multi-class classification

1 Introduction

Over the past few decades, there have been plenty of machine learning methods, such as association rules, K-nearest neighbor (KNN), artificial neural network and support vector machine (SVM)(Vapnik 1995). As a binary classifier, the SVM has excellent performance in solving small sample, nonlinear and high dimensional pattern recognition and has been widely applied in many fields. Extensive variants and extensions of SVMs have been studied in recent years, such as twin support vector machines (TSVM) (Jayadeva and Chandra 2007; An and Xue 2022), rough margin-based TSVMs (Xu et al 2012; Wang and Zhou 2017), weighted SVMs (Zhu et al 2016; Mir and Nasiri 2018), pinball SVMs (Huang et al 2014b; Rastogi et al 2018; Prasad and Balasundaram 2021), TPMSVMs (Sharma et al 2021), universum SVMs (Shi 2012; Qi et al 2014; Richhariya and Tanveer 2020), projection TSVRs (Peng and Chen 2018), robust TSVMs (Borah and Gupta 2021) and so on.

The standard SVM uses the hinge loss, which makes it sensitive to the presence of outliers and noises. As we know, the support vectors (SVs) play a crucial part in determining the separating hyperplane. The solution of SVM is sparse. In recent years, some techniques are used to improve the robustness of SVMs, such as using fuzzy theory (Jha and Mehta 2020; Rezvani and Wang 2021) and different loss functions (Bamakan et al 2017; Tang et al 2021). The fuzzy SVMs (Rezvani and Wang 2021) are robustness and can do uncertainty handling, and they can be applied to address the class imbalance issue in the presence of noise and outliers and large scale datasets.

✉ Zhijian Zhou
zhijianzh@163.com

[1] College of Science, Beijing Forestry University, No.35 Qinghua East Road, Haidian 100083, Beijing, China

[2] College of Science, China Agricultural University, No.17 Qinghua East Road, Haidian 100083, Beijing, China

The common used loss functions are square loss, Huber loss, pinball loss, and ramp loss, etc. The square loss is often adopted in least squares SVMs (LSSVMs) (Suykens and Vandewalle 1999; Kumar and Gopal 2009; Gupta and Richhariya 2018), which solves a system of linear equations instead of quadratic programming problems (QPPs) to obtain the optimal solution. Therefore, LSSVMs have fast solving speed. However, the solutions of LSSVMs totally lose sparsity. The Huber loss function (Borah and Gupta 2020) behaves as quadratic loss for a small fraction of closer points and grows linearly after a certain point. Therefore, it is a convex robust function. Finding the solutions of Huber loss-based SVMs is computational expensive for larger datasets. The pinball SVMs (Huang et al 2014b; Tanveer et al 2019c) are proposed to handle noise insensitivity and instability to re-sampling. It deals with the quantile distance rather than the closest distance of SVM, making it less sensitive to noise, especially feature noise. Not only the incorrectly classified samples gain penalties, but also the correctly classified samples gain penalties. Its pure form loses sparsity totally. To keep the sparsity as well as the noise insensitivity, some sparse SVMs with pinball loss (Tanveer et al 2019c, 2021b; Borah and Gupta 2021; Tang et al 2021) have been studied.

The direct reason for SVM is not robust enough is that the outliers gain the largest losses and are apt to become support vectors (SVs) (Bamakan et al 2017). The outliers will shift the hyperplane toward themselves inappropriately, thus leading to poor generalization ability. Considering the aforementioned issue, the ramp-loss-based SVM (RSVM) (Huang et al 2014a) was proposed to remedy this problem. The ramp loss function restricts the maximal loss value of the outliers, hence reduces their negative influences. What's more, the sparsity of traditional SVM is preserved. Although it makes the problem become a non-convex optimization problem, the Concave–Convex Programming (CCCP)(Yuille and Rangarajan 2003; Lipp and Boyd 2016) procedure can be applied to solve it by means of transforming it into a sequence of convex problems. With similar idea, some binary classifiers, such as the ramp nonparallel SVM (RNPSVM) (Liu et al 2015), the ramp LSSVM (Liu et al 2016), the ramp one-class SVM (Xiao et al 2017) and ramp maximum margin of TSVM (Lu et al 2019), have been proposed and studied.

The TSVM (Jayadeva and Chandra 2007; Tanveer et al 2019b) has attracted much attention for its faster computational speed because it solves two smaller sized QPPs. To mine the structural information of samples, some fuzzy SVMs (Deepak et al 2018; Rezvani and Wang 2021) have been researched which calculates the fuzzy membership for each sample. In order to exploit the underlying correlation or similarity information between the pair of samples with the same labels, a weighted TSVM (WTSVM) was proposed (Ye et al 2012). It has obtained comparable predictive accuracy compared with the TSVM. Some density-weighted SVMs (Hazarika and Gupta 2021) are also used to deal with noisy data classification. By integrating the rough set theory and WTSVM, Xu et al (2014) proposed a KNN-based weighted rough ν-TSVM in which not only different penalties are given according to the samples positions, but also different weights are given. Mir and Nasiri (2018) studied the least squares version of WTSVM which solves a pair of linear equations as opposed to solves two QPPs, thus increasing the solving speed. Gupta (2017) used the thought of WTSVM for solving regression problem.

Most of the aforementioned SVMs are binary classifiers, while the common classification scenes in the actual application are usually related to the multi-class classification problems. There are some effective decomposition strategies that can transform the multi-class problem into a series of binary problems (Hsu and Lin 2002; Balasundaram et al 2014). The common used methods include one-verse-one (OVO), one-verse-rest (OVR), all-versus-one (AVO) and direct acyclic graph (DAG) (Nasiri et al 2015; Ortigosa et al 2017; Zhou et al 2017; Kumar and Thakur 2018; Lu et al 2019). For a K-classification problem, the OVR method needs to construct K-binary classifiers. In each classifier, the j-class is regarded as positive class and the remaining is regarded as negative class, while AVO approach is just the opposite. These two approaches could easily lead to the imbalance problem and have bad performance. The OVO and DAG methods generate $K(K-1)/2$ binary classifiers, and each classifier involves only two classes of samples and ignores the remaining classes.

To mitigate the deficiencies of OVO and OVR methods, Angulo et al (2003) proposed a K-class support vector classification-regression (K-SVCR), which takes into account all the training points by adopting the "one-verse-one-verse-rest" (OVOVR) structure. It generates $K(K-1)/2$ binary classifiers in total and outputs the final predictive label by "voting" strategy. To increase its computational speed, the twin K-class support vector classification (TKSVC) (Xu et al 2013) was proposed, which has been proved to work approximately 4 times faster than the K-SVCR under the condition that the positive, the negative and rest samples are equal. The Ramp-TKSVC (Wang et al 2020) was proposed to reduce the negative influence of outliers and improve the robustness. Actually, the TKSVC and Ramp-TKSVC still work slowly, especially for large-scale multi-classification problem. A KNN-based weighted multi-class twin support vector machine (KWMTSVM) (Xu 2016) was proposed to improve the classification performance and computational efficiency by considering the local information of samples and removing the redundant constrains. Tanveer et al (2021a) proposed a least squares version of KWMTSVM (LS-KWMTSVM) which solves a pair of linear equations, thus accelerating the training process.

The KWMTSVM has gained promising results in tackling multi-classification problem. However, due to the convexity nature of the hinge loss function is used, it is susceptible to outliers in the training process. Because the outliers will gain the largest losses and easily become SVs, thus stretching the hyperplane toward themselves in an inappropriate way and leading to poor generalization ability. To circumvent this problem and develop a more precise, sparse and robust algorithm, we use the ramp loss to replace the hinge loss in the KWMTSVM and propose a ramp loss KNN-weighted multi-class twin support vector machine (RKWMTSVM) in this paper, and the contributions of this paper are listed as below:

- The proposed RKWMTSVM adopts OVOVR structure which can take a consideration of all training samples and overcome the data-imbalance problem.
- It restricts the loss of outlier to a fixed value and effectively suppresses the negative influence of constructing separating hyperplane, thus improving the robustness of the model.
- The outliers are avoided to become support vectors, then the proposed RKWMTSVM is sparser, especially compared to KWMTSVM.
- The proposed RKWMTSVM is a non-differentiable nonconvex optimization problem, we adopt the popular and handy CCCP method to effectively solve it. In each iteration of CCCP, it solves a sequence of KWMTSVM-like problems. The merits of KWMTSVM are inherited. That is to say, it can exploit the local information in terms of the data affinity simultaneously. By removing the redundant constraints, the RKWMTSVM has faster solving speed.
- To further accelerate the solving process of RKWMTS VM, the clipping dual coordinate descent (clipDCD) algorithm is utilized to solve the sub-quadratic programming problems in each iteration of CCCP.
- We do numerical experiments on twenty-four datasets and compare it with six state-of-the-art algorithms, i.e., OVO PGTSVM, OVR WTSVM, OVR RνTSVM, KWMTSVM, Ramp-TKSVC and LS-KWMTSVM. Exp erimental results verify the effectiveness of the proposed method.

The remainder of the paper is organized as follows. In Sect. 2, we outline the KWMTSVM and RSVM. In Sect. 3, we propose and analyze the novel algorithm named RKWMTSVM. In Sect. 4, we give the theoretical analysis on the proposed RKWMTSVM and discuss the proposed RKWMTSVM with six state-of-the-art algorithms. Section 5 performs the experiments to investigate the feasibility and validity of our proposed algorithm. In sect. 6, we introduce the clipDCD algorithm in our RKWMTSVM to increase the solving speed. Finally, we make conclusions in Sect. 7.

2 Related works

The SVM was initially proposed for binary classification problem. However, most of the problems in real life are multi-classification. To deal with K-class classification problem, two conventional approaches, i.e., OVO and OVR strategies are used. As mentioned above, the two approaches suffer from some drawbacks. Therefore, OVOVR structure was proposed to avoid the shortcomings of OVO and OVR.

By considering the training datasets as $S = \{(\mathbf{x}_1, y_1), ..., \}$ $\{(\mathbf{x}_l, y_l)\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ is the input vector, $y_i \in \{1, 2, ..., K\}$ is the corresponding label of $\mathbf{x}_i$, l and K are the number of samples and the number of classes, respectively. The KWMTSVM is an improved algorithm of TKSVC. They both need to construct $K(K-1)/2$ classifiers in total. In each classifier, it takes the OVOVR structure with ternary outputs $\{+1, -1, 0\}$, which considered all the training points. The i-class is considered as positive with label +1, the j-class is considered as negative with label -1, and the rest is considered as the zero class with label 0. Let matrix $\mathbf{A} \in \mathbb{R}^{l_1 \times n}$, $\mathbf{B} \in \mathbb{R}^{l_2 \times n}$, $\mathbf{C} \in \mathbb{R}^{l_3 \times n}$ stand for the positive, negative and zero class, respectively, and $l_1 + l_2 + l_3 = l$.

2.1 KNN-based weighted multi-class twin support vector machine (KWMTSVM)

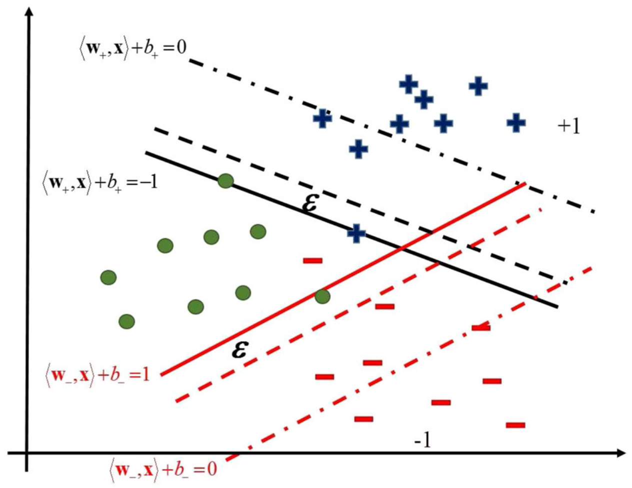
The KWMTSVM constructs the following pair of nonparallel hyperplanes:

$$\langle \mathbf{w}_+, \mathbf{x} \rangle + b_+ = 0 \quad \text{and} \quad \langle \mathbf{w}_-, \mathbf{x} \rangle + b_- = 0, \tag{1}$$

where $\mathbf{w}_+, \mathbf{w}_- \in \mathbb{R}^n$, $b_+, b_- \in \mathbb{R}$, such that each hyperplane is closer to one class and is as far as possible from the other. The remaining samples (zero class) are mapped into a region and satisfy two constraints $\langle \mathbf{w}_+, \mathbf{x} \rangle + b_+ \leq -1 + \varepsilon$ and $\langle \mathbf{w}_-, \mathbf{x} \rangle + b_- \geq 1 - \varepsilon$, where ε is a priori smaller scaler. It can be illustrated in Fig. 1, where the blue "+", red "−" and green "∘" stands for the positive, negative and zero class, respectively.

At the same time, when constructing the positive hyperplane, the KWMTSVM considered the local information of positive samples and exploited the information of inter-class by employing K-nearest neighbor. This thought originates the weighted TSVM (Ye et al 2012), which aims to discover the underlying similarity information within samples from the same class and can be modeled by the intra-class graph W_s. Simultaneously, it constructs a inter-class W_d to model the inter-class separability.

If the label of a sample $\mathbf{x}_i$ is $+1$, it needs to define two nearest sets: intra-class nearest sets $Nea_s(\mathbf{x}_i)$ which contains its neighbors in class $+1$ and inter-class nearest sets $Nea_d(\mathbf{x}_i)$ which contains its neighbors in the other class. Then, the two similarity matrices of graph W of class $+1$ corresponding



$\langle \mathbf{w}_+, \mathbf{x}\rangle + b_+ = 0$
$\langle \mathbf{w}_+, \mathbf{x}\rangle + b_+ = -1$
$\langle \mathbf{w}_-, \mathbf{x}\rangle + b_- = 1$
$\langle \mathbf{w}_-, \mathbf{x}\rangle + b_- = 0$
+1
-1
ε
ε

Fig. 1 The illustration of KWMTSVM

W_s and W_d can be defined as follows, respectively,

$$W_{s,ij} = \begin{cases} 1, & \text{if } \mathbf{x}_j \in Nea_s(\mathbf{x}_i) \text{ or } \mathbf{x}_i \in Nea_s(\mathbf{x}_j), \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

and

$$W_{d,ij} = \begin{cases} 1, & \text{if } \mathbf{x}_j \in Nea_d(\mathbf{x}_i), \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $Nea_s(\mathbf{x}_i) = \{\mathbf{x}_i^j | \text{if } \mathbf{x}_i^j \text{ and } \mathbf{x}_i \text{ belong to the same class}, 0 \le j \le k_1\}$, and $Nea_d(\mathbf{x}_i) = \{\mathbf{x}_i^j | \text{if } \mathbf{x}_i^j \text{ and } \mathbf{x}_i \text{ belong to diff erent classes}, 0 \le j \le k_2\}$.

Usually, d_j is used to measure the importance of sample, where

$$d_i = \sum_{j \in I_+} W_{s,ij}. \tag{4}$$

The bigger the value of d_i, the more "important" corresponding $\mathbf{x}_i$ is. In order to find the training samples which possibly become SVs in class -1, the weight matrices W_d can be redefined as follows,

$$f_j = \begin{cases} 1, & \text{if } \exists j, W_{d,ij} \neq 0, \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Similarly, the corresponding weight matrices of negative samples can be defined. In order to find the two hyperplanes, the KWMTSVM solves the following pair of quadratic programming problems (QPPs),

$$\min_{\mathbf{w}_+, b_+, \boldsymbol{\xi}, \boldsymbol{\eta}} \frac{1}{2} \sum_{i \in I_1} d_i (\mathbf{w}_+^T \mathbf{x}_i + b_+)^2 + c_1 \sum_{j \in I_2} \xi_j + c_2 \sum_{k \in I_3} \eta_k$$
$$\begin{aligned} s.t. \quad & -f_j(\mathbf{w}_+^T \mathbf{x}_j + b_+) + \xi_j \ge f_j,\ \xi_j \ge 0,\ j \in I_2, \\ & -h_k(\mathbf{w}_+^T \mathbf{x}_k + b_+) + \eta_j \ge (1-\varepsilon) h_k,\ \eta_k \ge 0,\ k \in I_3, \end{aligned} \tag{6}$$

and

$$\min_{\mathbf{w}_-, b_-, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*} \frac{1}{2} \sum_{j \in I_2} d_j (\mathbf{w}_-^T \mathbf{x}_j + b_-)^2 + c_3 \sum_{i \in I_1} \xi_i^* + c_4 \sum_{k \in I_3} \eta_k^*$$
$$\begin{aligned} s.t. \quad & f_i^*(\mathbf{w}_-^T \mathbf{x}_i + b_-) + \xi_i^* \ge f_i^*,\ \xi_i^* \ge 0,\ i \in I_1, \\ & h_k^*(\mathbf{w}_-^T \mathbf{x}_k + b_-) + \eta_k^* \ge (1-\varepsilon) h_k^*,\ \eta_k^* \ge 0,\ k \in I_3, \end{aligned} \tag{7}$$

where d_i and d_j are defined as (4), f_j, h_k, f_i^*, h_k^* are defined as (5), $\boldsymbol{\eta}, \boldsymbol{\eta}^* \in \mathbb{R}^{l_3}$, $\boldsymbol{\xi} \in \mathbb{R}^{l_2}$, $\boldsymbol{\xi}^* \in \mathbb{R}^{l_1}$, I_1, I_2 and I_3 are the indexes of samples belonging to the positive, negative and zero classes, respectively.

For a new testing sample $\mathbf{x}$, it can be predicted by the following decision function

$$f(\mathbf{x}) = \begin{cases} +1, & \text{if } \mathbf{w}_+^T \mathbf{x} + b_+ > -1 + \varepsilon \\ -1, & \text{if } \mathbf{w}_-^T \mathbf{x} + b_- < 1 - \varepsilon \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

2.2 Ramp-loss-based SVM

In the standard SVM, the hinge loss function is used by default, and it can be defined as $H_s(z) = max(0, s - z)$, where s is the position of the hinge point, is used to penalize those samples classified with an insufficient margin. The objective function of standard SVM with hinge loss function can be expressed as follows,

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{l} H_1(y_i f(\mathbf{x}_i)) \tag{9}$$

where $\mathbf{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$, and $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$ is the decision function.

The $H_s(z)$ is a convex loss function, and most SVMs with hinge loss function are sensitive to the presence of noises and outliers which will shift the decision hyperplane toward themselves inappropriately (Bamakan et al 2017). Because the outliers have the largest margin losses, the generalization ability of these algorithms decreases. Besides, these outliers will inevitably become the support vectors and make the algorithms have more computational cost. Recently, some non-convex losses are introduced the SVMs to make them more sparse and robust, and the ramp loss function attracts

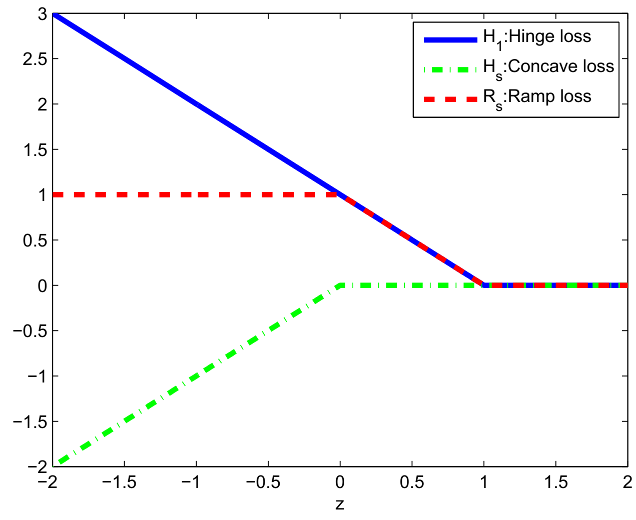


Fig. 2 The illustration of the ramp loss function, where $s = 0$

more attention for its better performance, where

$$R_s(z) = \begin{cases} 0, & z > 1 \\ 1 - z, & s \le z \le 1 \\ 1 - s, & z < s \end{cases} \tag{10}$$

where $s < 1$ is a prior given scalar. Fig. 2 clearly shows that the ramp loss is flat for scores z smaller than s, and the hinge loss function is a convex function. The $R_s(z)$ can be decomposed into the sum of a hinge loss and a concave loss, that is $R_s(z) = H_1(z) - H_s(z)$.

In order to increase the robustness of SVM and reduce the negative influence of outliers, the RSVM (Huang et al 2014a) has been proposed where the ramp loss function is used to replace the hinge loss function in SVM. Therefore, the primal problem of the RSVM can be formulated as follows,

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{l} R_s(y_i f(\mathbf{x}_i)) \\ &= \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{l} H_1(y_i f(\mathbf{x}_i)) - C\sum_{i=1}^{l} H_s(y_i f(\mathbf{x}_i)) \end{aligned} \tag{11}$$

This is a Concave–Convex optimization problem.

3 Ramp loss KNN-weighted multi-class twin support vector machine

Let $f_+(\mathbf{x}) = \mathbf{w}_+^T\mathbf{x} + b_+$, $f_-(\mathbf{x}) = \mathbf{w}_-^T\mathbf{x} + b_-$, the objective function of primal problems (6) and (7) is equivalent to

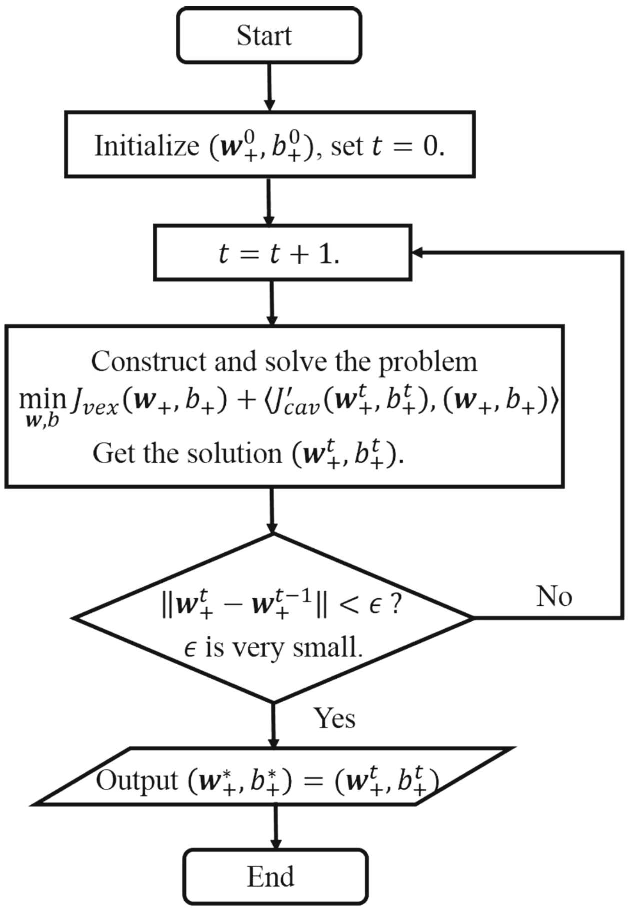

Start
Initialize $(\boldsymbol{w}_+^0, b_+^0)$, set $t = 0$.
$t = t + 1$.
Construct and solve the problem
$\min_{\boldsymbol{w},b} J_{vex}(\boldsymbol{w}_+, b_+) + \langle J'_{cav}(\boldsymbol{w}_+^t, b_+^t), (\boldsymbol{w}_+, b_+)\rangle$
Get the solution $(\boldsymbol{w}_+^t, b_+^t)$.
$\|\boldsymbol{w}_+^t - \boldsymbol{w}_+^{t-1}\| < \epsilon$?
ϵ is very small.
No
Yes
Output $(\boldsymbol{w}_+^*, b_+^*) = (\boldsymbol{w}_+^t, b_+^t)$
End

Fig. 3 The flowchart of CCCP for the problem (16)

$$\begin{aligned} &\min_{\mathbf{w}_+,b_+} \frac{1}{2}\sum_{i\in I_1} d_i(f_+(\mathbf{x}_i))^2 + c_1\sum_{j\in I_2} H_1(-f_j f_+(\mathbf{x}_j)) \\ &+ c_2\sum_{k\in I_3} H_{1-\varepsilon}(-h_k f_+(\mathbf{x}_k)), \end{aligned} \tag{12}$$

and

$$\begin{aligned} &\min_{\mathbf{w}_-,b_-} \frac{1}{2}\sum_{j\in I_2} d_j(f_-(\mathbf{x}_j))^2 + c_3\sum_{i\in I_1} H_1(f_i f_-(\mathbf{x}_i)) \\ &+ c_4\sum_{k\in I_3} H_{1-\varepsilon}(\tilde{h}_k f_-(\mathbf{x}_k)). \end{aligned} \tag{13}$$

From the two equations above and Fig. 2, the KWMTSVM adopts Hinge loss function which gives more punishments on those misclassified samples far from the hyperplane, making the hyperplane shifted toward the misclassified samples in an inappropriate way. Therefore, the KWMTSVM is sensitive to the presence of noises and outliers. It can be demonstrated by a toy example and is shown in Fig. 5. In Fig. 5a, the red and black thick solid line represent the decision boundary $f_+ = -1-\varepsilon$ and $f_- = 1+\varepsilon$, respectively. The "∘" and "△"

denote support vectors in QPP (6) and (7), respectively. The red sample located in the upper right corner is the noise of negative class. Obviously, it is circled by "∘" which means it becomes the support vector and shifts the positive hyperplane inappropriately. That means KWMTSVM overemphasizes the existence of noises, making it not robust enough. In order to further improve the robustness of the KWMTSVM, a new algorithm, termed as Ramp loss KNN-weighted multi-class twin support vector machine (RKWMTSVM), is proposed in the following.

3.1 Linear case

In Eq.(12), the second term and third term are the losses of samples in the negative class and zero class, respectively. These samples adopt the hinge loss function. We use the ramp loss function to replace them, and we can derive the first optimization problem of the proposed RKWMTSVM as follows,

$$\min_{\mathbf{w}_+,b_+} \frac{1}{2}\sum_{i\in I_1} d_i(f_+(\mathbf{x}_i))^2 + c_1\sum_{j\in I_2} R_{s_1}(-f_j f_+(\mathbf{x}_j)) + c_2\sum_{k\in I_3} R_{s_2}(-h_k f_+(\mathbf{x}_k)). \tag{14}$$

and similarly, the second optimization problem of the proposed RKWMTSVM is

$$\min_{\mathbf{w}_-,b_-} \frac{1}{2}\sum_{j\in I_2} d_j(f_-(\mathbf{x}_j))^2 + c_3\sum_{i\in I_1} R_{s_3}(f_i f_-(\mathbf{x}_i)) + c_4\sum_{k\in I_3} R_{s_4}(\tilde{h}_k f_-(\mathbf{x}_k)). \tag{15}$$

The ramp loss can be decomposed into the sum of convex hinge loss and concave function. Therefore, Eq.(14) and Eq.(15) can be reformulated as follows,

$$\min_{\mathbf{w}_+,b_+} \underbrace{\frac{1}{2}\sum_{i\in I_1} d_i(f_+(\mathbf{x}_i))^2 + c_1\sum_{j\in I_2} H_1(-f_j f_+(\mathbf{x}_j)) + c_2\sum_{k\in I_3} H_{1-\varepsilon}(-h_k f_+(\mathbf{x}_k))}_{J_{\mathrm{vex}}(\mathbf{w}_+,b_+)} \underbrace{-c_1\sum_{j\in I_2} H_{s_1}(-f_j f_+(\mathbf{x}_j)) - c_2\sum_{k\in I_3} H_{s_2}(-h_k f_+(\mathbf{x}_k)),}_{J_{\mathrm{cav}}(\mathbf{w}_+,b_+)} \tag{16}$$

and

$$\min_{\mathbf{w}_-,b_-} \underbrace{\frac{1}{2}\sum_{j\in I_2} d_j(f_-(\mathbf{x}_i))^2 + c_3\sum_{i\in I_1} H_1(f_i f_-(\mathbf{x}_i)) + c_4\sum_{k\in I_3} H_{1-\varepsilon}(h_k f_-(\mathbf{x}_j))}_{J_{\mathrm{vex}}(\mathbf{w}_-,b_-)} \underbrace{-c_3\sum_{i\in I_1} H_{s_3}(f_i f_-(\mathbf{x}_i)) - c_4\sum_{k\in I_3} H_{s_4}(\tilde{h}_k f_-(\mathbf{x}_k))\,.}_{J_{\mathrm{cav}}(\mathbf{w}_-,b_-)} \tag{17}$$

Obviously, the above two problems consist of two parts: the convex part J_{vex} and the concave part J_{cav}, and they need to minimize a non-convex cost function. The CCCP is a powerful heuristic method which is used to find local solutions to difference of convex (DC) programming problems (Yuille and Rangarajan 2003; Lipp and Boyd 2016). Because it is simple in tuning and can iteratively solve a sequence of convex programs, it has become a successful and power algorithm for solving non-differentiable non-convex optimization problems (Bamakan et al 2017). For simplicity and convenience, we only use problem (16) of the proposed RKWMTSVM to interpret the detailed solving progress, and the problem (17) can be obtained with an exactly similar way.

The CCCP framework for problem (16) is constructed in the following flowchart, seen (Fig. 3).

From the flowchart, the CCCP procedure for problem (16) needs to solve the following problem in the t-th step,

$$\min_{\mathbf{w}_+,b_+} J_{\mathrm{vex}}(\mathbf{w}_+,b_+) + \left\langle J'_{\mathrm{cav}}(\mathbf{w}_+^t,b_+^t),(\mathbf{w}_+,b_+)\right\rangle \tag{18}$$

Next, we will show how to find the solution for Eq.(18). Remarkably, $J'_{\mathrm{cav}}(\mathbf{w}_+^t,b_+^t)$ is non-differentiable at some points. When using any upper-derivative of a concave function, the CCCP can remain valid. For convenience, we suppose $\boldsymbol{\delta}_+ = (\delta_1,\delta_2,...,\delta_{l_2})^T$, $\boldsymbol{\theta}_+ = (\theta_1,\theta_2,...,\theta_{l_3})^T$, where

$$\delta_j = -c_1\cdot\frac{\partial H_{s_1}(-f_j f_+(\mathbf{x}_j))}{\partial(-f_j f_+(\mathbf{x}_j))}$$

$$= \begin{cases} c_1, & \text{if } -f_j f_+(\mathbf{x}_j) < s_1, f_j = 1, \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

where $j \in I_2$, and

$$\theta_k = -c_2 \cdot \frac{\partial H_{s_2}(-h_k f_+(\mathbf{x}_k))}{\partial(-h_k f_+(\mathbf{x}_j))} = \begin{cases} c_2, & \text{if } -h_k f_+(\mathbf{x}_k) < s_2, h_k = 1, \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

where $k \in I_3$. Therefore, using Eq.(19) and Eq.(20), Eq. (18) can be rewritten as:

$$\min_{\mathbf{w}_+,b_+} \frac{1}{2}\sum_{i\in I_1} d_i(\mathbf{w}_+^T\mathbf{x}_i + b_+)^2 + c_1\sum_{j\in I_2} H_1(-f_j f_+(\mathbf{x}_j)) + c_2\sum_{k\in I_3} H_{1-\varepsilon}(-h_k f_+(\mathbf{x}_k)) - \sum_{j\in I_2}\delta_j f_j(\mathbf{w}_+^T\mathbf{x}_j + b_+) - \sum_{k\in I_3}\theta_k h_k(\mathbf{w}_+^T\mathbf{x}_k + b_+), \tag{21}$$

where the first three items are the convex part $J_{\text{vex}}(\mathbf{w}_+, b_+)$ in Eq. (18), the fourth item is the result of $-c_1$ multiplies the inner product of $\frac{\partial H_{s_1}(-f_j f_+(\mathbf{x}_j))}{\partial(-f_j f_+(\mathbf{x}_j))}$ and $-f_j f_+(\mathbf{x}_j)$ $(f_+(\mathbf{x}_j) = \mathbf{w}_+^T\mathbf{x}_j + b_+)$, and the last term is the result of $-c_2$ multiplies the inner product of $\frac{\partial H_{s_2}(-h_k f_+(\mathbf{x}_k))}{\partial(-h_k f_+(\mathbf{x}_k))}$ and $-f_k f_+(\mathbf{x}_k)$ $(f_+(\mathbf{x}_k) = \mathbf{w}_+^T\mathbf{x}_k + b_+)$.

By introducing slack vectors $\boldsymbol{\xi}_+ = (\xi_1, \xi_2, ..., \xi_{l_2})^T$ and $\boldsymbol{\eta}_+ = (\eta_1, \eta_2, ..., \eta_{l_3})^T$, the above equation is equivalent to

$$\begin{aligned} \min_{\mathbf{w}_+,b_+} \ & \frac{1}{2}\sum_{i\in I_1} d_i(\mathbf{w}_+^T\mathbf{x}_i + b_+)^2 + c_1\sum_{j\in I_2}\xi_j + c_2\sum_{k\in I_3}\eta_k \\ & - \sum_{j\in I_2}\delta_j f_j(\mathbf{w}_+^T\mathbf{x}_j + b_+) - \sum_{k\in I_3}\theta_k h_k(\mathbf{w}_+^T\mathbf{x}_k + b_+) \\ s.t. \ & -f_j(\mathbf{w}_+^T\mathbf{x}_j + b_+) + \xi_j \geq f_j, \xi_j \geq 0, j \in I_2, \\ & -h_k(\mathbf{w}_+^T\mathbf{x}_k + b_+) + \eta_k \geq (1-\varepsilon)h_k, \eta_k \geq 0, k \in I_3. \end{aligned} \tag{22}$$

We can construct the Lagrangian function of (22) as follows,

$$\begin{aligned} & L(\mathbf{w}_+, b_+, \boldsymbol{\xi}_+, \boldsymbol{\eta}_+, \boldsymbol{\alpha}_+, \boldsymbol{\beta}_+, \boldsymbol{\lambda}_+, \boldsymbol{\sigma}_+) \\ & = \frac{1}{2}\sum_{i\in I_1} d_i(\mathbf{w}_+^T\mathbf{x}_i + b_+)^2 + c_1\sum_{j\in I_2}\xi_j + c_2\sum_{k\in I_3}\eta_k \\ & - \sum_{j\in I_2}\delta_j f_j(\mathbf{w}_+^T\mathbf{x}_j + b_+) - \sum_{k\in I_3}\theta_k h_k(\mathbf{w}_+^T\mathbf{x}_k + b_+) \\ & - \sum_{j\in I_2}\alpha_j^+(-f_j(\mathbf{w}_+^T\mathbf{x}_j + b_+) + \xi_j - f_j) \\ & - \sum_{k\in I_3}\beta_k^+(-h_k(\mathbf{w}_+^T\mathbf{x}_k + b_+) + \eta_k - (1-\varepsilon)h_k) \\ & - \sum_{j\in I_2}\lambda_j^+\xi_j - \sum_{k\in I_3}\sigma_k^+\eta_k, \end{aligned} \tag{23}$$

where $\boldsymbol{\alpha}_+ = (\alpha_j^+)_{i\in I_2}$, $\boldsymbol{\beta}_+ = (\beta_k^+)_{k\in I_3}$, $\boldsymbol{\lambda} = (\lambda_j^+)_{j\in I_2}$, and $\boldsymbol{\sigma}_+ = (\sigma_k^+)_{k\in I_3}$ are Lagrange multipliers. Differentiating the Lagrangian function L with respect to variables $\mathbf{w}_+, b_+, \boldsymbol{\xi}_+, \boldsymbol{\eta}_+$ can yield the following Karush–Kuhn–Tucker (KKT) conditions:

$$\frac{\partial L}{\partial \mathbf{w}_+} = \sum_{i\in I_1} d_i(\mathbf{x}_i\mathbf{x}_i^T\mathbf{w}_+ + \mathbf{x}_i b_+) + \sum_{j\in I_2}\alpha_j^+ f_j\mathbf{x}_j + \sum_{k\in I_3}\beta_k^+ h_k\mathbf{x}_k - \sum_{j\in I_2}\delta_j f_j\mathbf{x}_j - \sum_{k\in I_3}\theta_k h_k\mathbf{x}_k = 0, \tag{24}$$

$$\frac{\partial L}{\partial b_+} = \sum_{i\in I_1} d_i(\mathbf{x}_i^T\mathbf{w}_+ + b_+) + \sum_{j\in I_2}\alpha_j^+ f_j + \sum_{k\in I_3}\beta_k^+ h_k - \sum_{j\in I_2}\delta_j f_j - \sum_{k\in I_3}\theta_k h_k = 0, \tag{25}$$

$$\frac{\partial L}{\partial \boldsymbol{\xi}_+} = c_1\mathbf{e}_2 - \boldsymbol{\alpha}_+ - \boldsymbol{\lambda}_+ = 0, \tag{26}$$

$$\frac{\partial L}{\partial \boldsymbol{\eta}_+} = c_2\mathbf{e}_3 - \boldsymbol{\beta}_+ - \boldsymbol{\sigma}_+ = 0, \tag{27}$$

$$\sum_{j\in I_2}\alpha_j^+(-f_j(\mathbf{w}_+^T\mathbf{x}_j + b_+) + \xi_j - f_j) = 0, \tag{28}$$

$$\sum_{k\in I_3}\beta_k^+(-h_k(\mathbf{w}_+^T\mathbf{x}_k + b_+) + \eta_k - (1-\varepsilon)h_k) = 0, \tag{29}$$

$$\sum_{j\in I_2}\lambda_j^+\xi_j = 0, \sum_{k\in I_3}\sigma_k^+\eta_k = 0. \tag{30}$$

Arranging Eqs.(24) and (25) and representing them in the matrix form, we can get

$$\mathbf{A}^T\mathbf{D}_1(\mathbf{A}\mathbf{w}_+ + \mathbf{e}_1 b_+) + \mathbf{B}^T\mathbf{F}_2(\boldsymbol{\alpha}_+ - \boldsymbol{\delta}_+) + \mathbf{C}^T\mathbf{H}(\boldsymbol{\beta}_+ - \boldsymbol{\theta}_+) = 0, \tag{31}$$

$$\mathbf{e}_1^T\mathbf{D}_1(\mathbf{A}\mathbf{w}_+ + \mathbf{e}_1 b_+) + \mathbf{e}_2^T\mathbf{F}_2(\boldsymbol{\alpha}_+ - \boldsymbol{\delta}_+) + \mathbf{e}_3^T\mathbf{H}(\boldsymbol{\beta}_+ - \boldsymbol{\theta}_+) = 0, \tag{32}$$

where $\mathbf{D}_1 = diag(d_1, d_2, ..., d_{l_1})$, $\mathbf{F}_2 = diag(f_1, f_2, ..., f_{l_2})$, $\mathbf{H} = diag(h_1, h_2, ..., h_{l_3})$, $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ are vectors of ones of appropriate dimensions.

By combining Eqs.(31) and (32) and setting $\mathbf{u}_+ = [\mathbf{w}_+;\ b_+]$, we get

$$\mathbf{u}_+ = -(\mathbf{X}_1^T\mathbf{D}_1\mathbf{X}_1)^{-1}(\mathbf{X}_2^T\mathbf{F}_2(\boldsymbol{\alpha}_+ - \boldsymbol{\delta}_+) + \mathbf{X}_3^T\mathbf{H}(\boldsymbol{\beta}_+ - \boldsymbol{\theta}_+)), \tag{33}$$

where $\mathbf{X}_1 = [\mathbf{A}\ \mathbf{e}_1]$, $\mathbf{X}_2 = [\mathbf{B}\ \mathbf{e}_2]$, and $\mathbf{X}_3 = [\mathbf{C}\ \mathbf{e}_3]$.

From (26) and (27), we have

$$\mathbf{0} \le \boldsymbol{\alpha}_+ \le c_1\mathbf{e}_2, \tag{34}$$

$$\mathbf{0} \le \boldsymbol{\beta}_+ \le c_2\mathbf{e}_3, \tag{35}$$

where $\mathbf{0}$ is vector of zeros of appropriate dimension. For simplicity, we set $\boldsymbol{\gamma}_+ = [\boldsymbol{\alpha}_+;\ \boldsymbol{\beta}_+]$, $\boldsymbol{\tau}_+ = [\boldsymbol{\delta}_+;\ \boldsymbol{\theta}_+]$, $\mathbf{N}_+ = [\mathbf{X}_2^T\mathbf{F}_2;\ \mathbf{X}_3^T\mathbf{H}]$, $\mathbf{e}_4 = [\mathbf{F}_2\mathbf{e}_2;\ \mathbf{H}\mathbf{e}_3(1-\varepsilon)]$, and $\mathbf{S}_+ = [c_1\mathbf{e}_2;\ c_2\mathbf{e}_3]$.

By substituting the equations above into (23), we can derive the dual formulation as follows,

$$\max_{\boldsymbol{\gamma}_+} \ -\frac{1}{2}(\boldsymbol{\gamma}_+ - \boldsymbol{\tau}_+)^T\mathbf{N}_+(\mathbf{X}_1^T\mathbf{D}_1\mathbf{X}_1)^{-1}\mathbf{N}_+(\boldsymbol{\gamma}_+ - \boldsymbol{\tau}_+) + \mathbf{e}_4^T\boldsymbol{\gamma}_+$$
$$s.t.\ \ \mathbf{0} \le \boldsymbol{\gamma}_+ \le \mathbf{S}_+. \tag{36}$$

Similarly, the dual of the QPP (17) is derived as

$$\max_{\boldsymbol{\gamma}_-} \ -\frac{1}{2}(\boldsymbol{\gamma}_- - \boldsymbol{\tau}_-)^T\mathbf{N}_-(\mathbf{X}_2^T\mathbf{D}_2\mathbf{X}_2)^{-1}\mathbf{N}_-(\boldsymbol{\gamma}_- - \boldsymbol{\tau}_-) + \mathbf{e}_5^T\boldsymbol{\gamma}_-$$
$$s.t.\ \ \mathbf{0} \le \boldsymbol{\gamma}_- \le \mathbf{S}_-, \tag{37}$$

where $\mathbf{D}_2 = diag(d_1, d_2, ..., d_{l_2})$, $\mathbf{F}_1 = diag(f_1, f_2, ..., f_{l_1})$, $\mathbf{H}^* = diag(h_1, h_2, ..., h_{l_3})$, $\mathbf{N}_- = [\mathbf{X}_1^T\mathbf{F}_1;\ \mathbf{X}_3^T\mathbf{H}^*]$, $\mathbf{e}_5 = [\mathbf{F}_1\mathbf{e}_1;\ \mathbf{H}^*\mathbf{e}_3(1-\varepsilon)]$, and $\mathbf{S}_- = [c_3\mathbf{e}_1;\ c_4\mathbf{e}_3]$. Similarly, we set $\boldsymbol{\gamma}_- = [\boldsymbol{\alpha}_-;\ \boldsymbol{\beta}_-]$, where $\boldsymbol{\alpha}_-, \boldsymbol{\beta}_-$ are Lagrange multipliers, and set $\boldsymbol{\tau}_- = [\boldsymbol{\delta}_-;\ \boldsymbol{\theta}_-]$, where $\boldsymbol{\delta}_- = (\bar{\delta}_1, \bar{\delta}_2, ..., \bar{\delta}_{l_1})^T$, $\boldsymbol{\theta}_- = (\bar{\theta}_1, \bar{\theta}_2, ..., \bar{\theta}_{l_3})^T$, and

$$\bar{\delta}_j = -c_3 \cdot \frac{\partial H_{s_3}(f_-(\mathbf{x}_i))}{\partial(f_-(\mathbf{x}_i))} = \begin{cases} c_3, & \text{if } f_i f_-(\mathbf{x}_i) < s_3,\ f_i = 1, \\ 0, & \text{otherwise} \end{cases} \tag{38}$$

where $i \in I_1$, and

$$\bar{\theta}_k = -c_4 \cdot \frac{\partial H_{s_4}(f_-(\tilde{h}_k\mathbf{x}_k))}{\partial(\tilde{h}_k f_-(\mathbf{x}_k))} = \begin{cases} c_4, & \text{if } \tilde{h}_k f_-(\mathbf{x}_k) < s_4,\ \tilde{h}_k = 1, \\ 0, & \text{otherwise} \end{cases} \tag{39}$$

where $k \in I_3$.

Once the optimal solution $\boldsymbol{\gamma}_-$ is obtained, we can derive

$$\mathbf{u}_- = (\mathbf{X}_2^T\mathbf{D}_2\mathbf{X}_2)^{-1}(\mathbf{X}_1^T\mathbf{F}_1(\boldsymbol{\alpha}_- - \boldsymbol{\delta}_-) + \mathbf{X}_3^T\mathbf{H}^*(\boldsymbol{\beta}_- - \boldsymbol{\theta}_-)), \tag{40}$$

where $\mathbf{u}_- = [\mathbf{w}_-;\ b_-]$.

Based on the aforementioned equations, the algorithm of linear RKWTSVM based on the CCCP procedure is summarized as follows.

3.2 Nonlinear case

In an exactly similar way, we extend the linear RKWTSVM to the nonlinear case by introducing the kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = (\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j))$. The nonlinear case differs from the linear case only in that

Algorithm 1 CCCP for the proposed RKWMTSVM.

```
Input: The training dataset $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l)\}$, where label
  $y_i \in 1, 2, ..., K$, and the testing sample $\mathbf{x}$.
Output: The label of testing sample.
  1. Choose the appropriate $s_i < 1, i = 1, ..., 4$ for the ramp loss
  $R_s(z)$, parameters $\varepsilon \in (0, 1)$ and $c_i > 0, i = 1, ..., 4$.
  2. For each pair $(i, j) \in \{(i, j)|i < j, i, j = 1, ..., K\}$, divide the
  training dataset into three parts: positive (class $i$), negative (class $j$)
  and zero class (the rest).
    1) Initialize $(\boldsymbol{\delta}_+^0, \boldsymbol{\theta}_+^0)$ and $(\boldsymbol{\delta}_-^0, \boldsymbol{\theta}_-^0)$, and set $t = 0$.
    2) Construct and solve the dual problems (36) and (37) in the $t$-
       th iteration step to obtain the optimal solutions $(\boldsymbol{\alpha}_+^t, \boldsymbol{\beta}_+^t)$ and
       $(\boldsymbol{\alpha}_-^t, \boldsymbol{\beta}_-^t)$, then calculate $(\mathbf{w}_+^t, b_+^t)$ and $(\mathbf{w}_-^t, b_-^t)$ according to
       (33) and (40), respectively. Compute the decision functions $f_+^t(\mathbf{x})$
       and $f_-^t(\mathbf{x})$.
    3) Update $(\boldsymbol{\delta}_+^t, \boldsymbol{\theta}_+^t)$ and $(\boldsymbol{\delta}_-^t, \boldsymbol{\theta}_-^t)$ referring to (19), (20), (38) and
       (39), respectively.
    4) If $(\boldsymbol{\delta}_+^t, \boldsymbol{\theta}_+^t) = (\boldsymbol{\delta}_+^{t-1}, \boldsymbol{\theta}_+^{t-1})$ and $(\boldsymbol{\delta}_-^t, \boldsymbol{\theta}_-^t) = (\boldsymbol{\delta}_-^{t-1}, \boldsymbol{\theta}_-^{t-1})$, stop the
       iteration and obtain the optimal solutions $(\mathbf{w}_+^*, b_+^*) = (\mathbf{w}_+^t, b_+^t)$,
       and $(\mathbf{w}_-^*, b_-^*) = (\mathbf{w}_-^t, b_-^t)$. Otherwise, set $t = t + 1$, and go to
       the step 2).
  3. For a new testing point $\mathbf{x}$, predict its label according to the decision
  functions $f_{(i,j)}(\mathbf{x})$, $(i, j) \in \{(i, j)|i < j, i, j = 1, ..., K\}$, and
  output its label by "voting" strategy.
```

(1) The matrices $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$ in (36) and (37) are instead of the appropriate kernel forms $\mathbf{X}_1 = [K(\mathbf{A}, \mathbf{D})\ \mathbf{e}_1]$, $\mathbf{X}_2 = [K(\mathbf{B}, \mathbf{D})\ \mathbf{e}_2]$, and $\mathbf{X}_3 = [K(\mathbf{C}, \mathbf{D})\ \mathbf{e}_3]$, where $\mathbf{D} = [\mathbf{A}; \mathbf{B}; \mathbf{C}]$.

(2) The decision function becomes the following equation,

$$f(\mathbf{x}) = \begin{cases} +1, & \text{if } \mathbf{w}_+^T K(\mathbf{x}^T, \mathbf{D}) + b_+ > -1 + \varepsilon \\ -1, & \text{if } \mathbf{w}_-^T K(\mathbf{x}^T, \mathbf{D}) + b_- < 1 - \varepsilon \\ 0, & \text{otherwise} \end{cases} \tag{41}$$

4 Theoretical analysis and comparison

We do theoretical analysis on RKWMTSVM in terms of sparsity and algorithm analysis and compare it with state-of-the-art algorithms.

4.1 Sparsity

Compared with KWMTSVM, the proposed RKWMTSVM is sparser. Because of the introduction of ramp loss, all misclassified samples are avoided converting to SVs in the RKWMTSVM. We assume that the Ramp losses $R_{s_i}, i = 1, ..., 4$ are made differentiable with smooth approximation on a small interval of corresponding inflection samples. Differentiating (14) and (15), we can obtain

$$\sum_{i\in I_1} d_i(\mathbf{w}_+^T\mathbf{x}_i + b_+)\mathbf{x}_i = c_1 \sum_{j\in I_2} f_j R'_{s_1}(-f_j f_+(\mathbf{x}_j))\mathbf{x}_j$$

$$+ c_2 \sum_{k\in I_3} h_k R'_{s_2}(-h_k f_+(\mathbf{x}_k))\mathbf{x}_k, \tag{42}$$

$$\sum_{j\in I_2} d_j(\mathbf{w}_-^T\mathbf{x}_j + b_-)\mathbf{x}_j = -c_3 \sum_{i\in I_1} f_i R'_{s_3}(f_i f_-(\mathbf{x}_i))\mathbf{x}_i - c_4 \sum_{k\in I_3} \tilde{h}_k R'_{s_4}(\tilde{h}_k f_-(\mathbf{x}_k))\mathbf{x}_k, \tag{43}$$

respectively.

From Eq.(42), we can conclude that the negative samples $\mathbf{x}_j$ located in the flat area, i.e., $z = -f_j f_+(\mathbf{x}_j) \notin [s_1, 1]$, will not become SVs for the reason that $R'_{s_1} = 0$; the zero samples $\mathbf{x}_k$ in the flat area, i.e., $z = -h_k f_+(\mathbf{x}_k) \notin [s_2, 1-\varepsilon]$, will not become SVs because $R'_{s_2} = 0$. We can do similar analysis for Eq.(43). Compared with KWMTSVM, the proposed RKWMTSVM can explicitly incorporate outlier suppression better. Therefore, RKWMTSVM is more robust and sparser.

The analysis above can be proved by the following property.

Sparsity. From Eq. (33), the support vectors are the corresponding negative samples with $\alpha_j^+ - \delta_j \neq 0$, for $j \in I_2$ and the corresponding "rest" samples with $\beta_j^+ - \theta_j \neq 0$, for $j \in I_3$. Then, the following statements can be derived.

(1) If $\alpha_j^+ = c_1$, we get $\delta_j = c_1$ from (19), therefore, $\alpha_j^+ - \delta_j = 0$, for $j \in I_2$.
(2) If $\alpha_j^+ = 0$, we get $\delta_j = 0$ from (19), therefore, $\alpha_j^+ - \delta_j = 0$, for $j \in I_2$.
(3) If $\beta_k^+ = c_2$, we get $\theta_k = c_2$ from (20), therefore, $\beta_k^+ - \theta_k = 0$, for $k \in I_3$.
(4) If $\beta_k^+ = 0$, we get $\theta_k = 0$ from (20), therefore, $\beta_k^+ - \theta_k = 0$, for $k \in I_3$.

From the statements above, the bounded SVs (with $\alpha_j^+ = c_1, j \in I_2$ or $\beta_k^+ = c_2, k \in I_3$) in the KWMTSVM are not SVs any more in the proposed RKWMTSVM. Similarly, we can derive the corresponding statements from Eq. (40) and obtain the similar conclusions. Consequently, we conclude that the proposed RKWMTSVM is more sparse than KWMTSVM.

In the following, we analyze the similarities and differences of the proposed RKWMTSVM with five sparse twin SVMs. They all need to construct two non-parallel hyperplanes.

(1) STSVM (Peng 2011a). Their similarity is that their optimal solutions are obtained by iteration. The STSVM is proposed for binary classification problem, and it is built in primal space, while RKWMTSVM is proposed for multi-class classification problem and it is solved in the dual space. Based on a simple back-fitting strategy, the STSVM iteratively builds each nonparallel hyperplanes by adding a new SV from the corresponding class at one time, and its learning style is solving linear equation computing systems. The RKWMTSVM iteratively solves a sequence of KWMTSVM-like optimization problems, and KWMTSVM is a convex optimization problem.
(2) SNPTSVM (Tian et al 2014). The SNPTSVM is proposed for binary classification problem. Its sparsity is achieved by adopting the ϵ-insensitive quadratic loss function and soft margin quadratic loss function. For the positive hyperplane, the SNPTSVM can achieve sparsity for positive samples while RKWMTSVM can not, and the sparsity of RKWMTSVM is mainly reflected in the negative and zero samples. However, the SNPTSVM sacrifices the complexity because its dual problem contains three kinds of Lagrangian multipliers.
(3) SPinTSVM (Tanveer et al 2019a). The SPinTSVM is raised for binary classification problem. It achieves the sparsity by using the ϵ-insensitive zone pinball loss function. Not only the incorrectly classified samples gain punishments, but also the correctly classified samples gain penalties. It is less sensitive to noise, especially feature noise. The proposed RKWMSTVM achieves sparsity by using the ramp loss function. It only punishes the incorrectly classified data and when the outlier is bigger than a threshold s, the corresponding penalty will be given a fixed value. Therefore, the RKWMSTVM is less sensitive to label noise.
(4) RSLPTSVM (Tanveer 2014). The RSLPTSVM is a binary classifier while RKWMTSVM is a multi-class classification algorithm. The RSLPTSVM achieves sparsity by using exact 1-norm, and it is a strongly convex problem by incorporated regularization term while RKWMSTVM is a non-convex optimization algorithm. The optimal solution of RSLPTSVM is obtained by solved by a pair of unconstrained minimization problems using Newton–Armijo algorithm while RKWMTSVM solves a series of QPPs.
(5) RNPSVM (Liu et al 2015). What RNPSVM and RKWM TSVM have in common is that they both use ramp loss function which can explicitly incorporate noise and outlier suppression in the training phase. They both are non-convex and non-differentiable optimization problems and have sparsity and robustness. The difference between them is that RKWMTSVM is proposed for multi-classification by adopting OVOVR structure, while RNPSVM is proposed for binary classification. Besides, RNPSVM preserves the ϵ-insensitive function of NPSVM. RKWMTSVM excavates the structural information of data to improve the generalization ability and accelerate the solving speed.
(6) R-TWSVM (Qi et al 2013). The R-TWSVM is a stronger insensitive binary classifier for dealing with missing

or uncertain data with label noise. Because it uses a quadratic loss function making the proximal hyperplane close to the class itself, and a soft-margin loss function making the hyperplane as far as possible from the other class. The RKWMTSVM can effectively depress the negative influence of outliers in the negative (resp. positive) and zero classes for the positive (resp. negative) classifier by using ramp loss functions. The R-TWSVM is a second-order cone programming problem, while RKWMTSVM is a non-convex optimization problem.

4.2 Algorithm analysis

The proposed RKWMTSVM aims at constructing two non-parallel hyperplanes by solving a pair of smaller-size QPPs in each sub-classifier.

- When constructing the separating positive (negative) hyperplane, the greater weights are given to the positive (negative) samples if they have more KNNs.
- The ineffective constraints can be removed if their corresponding KNN weights $f_j = 0, j \in I_2$ (resp. $f_i = 0, i \in I_1$) and $h_k = 0, k \in I_3$ (resp. $\tilde{h}_k = 0, k \in I_3$), thus accelerating the computational speed of RKWMTSVM.
- The structural information is exploited by KNN, and the class imbalance problem is resolved by adopting the OVOVR structure.
- The KWMTSVM can be seen as a special case of the proposed RKWMTSVM. If $s_i \to -\infty, i = 1, ..., 4$, then $R_s \to H_1$, i.e., s takes large negative value, the ramp loss in the RKWMTSVM will not help to remove outliers, then RKWMTSVM will degenerate to KWMTSVM.
- Computational complexity of RKWMTSVM. In a 3-class classification problem, we suppose that there are $l/3$ samples in each class, where l is the number of total number of training samples. As discussed in the works proposed by Xu (2016) and Tanveer et al (2021a), the computational complexity of TKSVC $O(2 \times (2l/3)^3)$. Because of $f_j = 1$ or $0, j \in I_2$ (resp. $f_i = 1$ or $0, i \in I_1$) and $h_k = 1$ or $0, k \in I_3$ (resp. $\tilde{h}_k = 1$ or $0, k \in I_3$), the computational complexity of KWMTSVM is less than $O(16l^3/27)$. The proposed RKWMTSVM needs to solve a sequence of KWMTSVM-like problems; therefore, the computational complexity of RKWMTSVM is less than $O(16l^3/27)$ times the iteration number in CCCP.

The proposed RKWMTSVM still needs to calculate the weight matrices by KNN steps whose complexity is $O(l^2\log(l))$. Therefore, the total time complexity of RKWMTSVM is about $O(l^2\log(l))$ plus $O(16l^3/27)$ times the iteration number.

4.3 Discussion on RKWMTSVM

4.3.1 OVO PGTSVM vs. RKWMTSVM

The PGTSVM (Tanveer et al 2019c) is noise insensitive and more stable for re-sampling because it uses the pinball loss function, which not only gives punishments on correctly classified samples but also on incorrectly classified samples. Besides, the pinball loss function does not increase the computational complexity as compared with TSVM. Compared with RKWMTSVM, it totally loses sparsity.

Because PGTSVM is proposed for binary classification, we extend it to multi-classification case by adopting OVO strategy, termed as OVO PGTSVM. For a 3-class classification problem, we still set there are $l/3$ samples in each class (The same for the following analysis). OVO PGTSVM has to construct $K(K-1)/2$ binary PGTSVM, and each binary PGTSVM only picks two classes. Therefore, its computational complexity is about $O(2 \times (l/3)^3)$.

4.3.2 OVR WTSVM vs. RKWMTSVM

The WTSVM (Ye et al 2012) is a binary classifier which can mine the underlying similarity information within samples by using inner-class nearest neighbor, which greatly improve the performance. Besides, it removes the redundant constraints by using inter-class nearest neighbor to improve the solving speed. WTSVM is the basic algorithm compared with RKWMTSVM, because RKWMTSVM also considers inner-class and inter-class nearest neighbor to excavate the structural information.

The OVR is a common used "decomposition-reconstruction" strategy. We extend WTSVM to multi-classification case by employing OVR, termed as OVR WTSVM. For a 3-class classification problem, OVR WTSVM needs to construct K binary WTSVMs, and the computational complexity of each binary WTSVM is less than $O(2 \times (2l/3)^3)$. Besides, it needs to calculate k-nearest neighbor graph, whose computational complexity is about $O(l^2\log(l))$.

4.3.3 OVR RνTSVM vs. RKWMTSVM

Rough ν-TSVM (Rν-TSVM) (Xu et al 2012) is proposed as a binary classifier, which gives different penalties to different misclassified samples according to their positions by constructing rough lower margin, rough upper margin and rough boundary. It can avoid the over-fitting problem to a certain extent. Both Rν-TSVM and RKWMTSVM concentrate on the study of misclassified samples. The RKWMTSVM uses ramp loss function to depress the negative influence of outliers which are also the misclassified samples.

We extend Rν-TSVM to multi-classification scenes by OVR. For a 3-class classification problem, OVR Rν-TSVM also has to construct K binary Rν-TSVMs, and the computational complexity of each binary Rν-TSVM is about $O(2 \times (2l/3)^3)$.

4.3.4 KWMTSVM vs. RKWMTSVM

Both KWMTSVM (Xu 2016) and RKWMTSVM adopt OVOVR structure to handle multi-classification data and consider the local information of samples by KNN. The difference of them is the usage of loss function, where KWMTSVM uses the hinge loss function while the proposed RKWMTSVM uses the ramp loss function. The KWMTSVM solves a pair of convex optimization problems, while RKWMTSVM is a concave–convex optimization problem. The RKWMTSVM solves a sequence of KWMTSVM-like problems by CCCP. Besides, RKWMTS VM is more robust for outliers. Compared with KWMTSVM, our RKWMTSVM is sparser.

For a 3-class classification problem, KWMTSVM needs to solve two QPPs and use KNN steps to compute weight matrices. Its overall time complexity is approximately $O(l^2\log(l) + 16l^3/27)$.

4.3.5 Ramp-TKSVC vs. RKWMTSVM

Ramp-TKSVC (Wang et al 2020) is also a multi-classification model adopting OVOVR structure. Both Ramp-TKSVC and RKWMTSVM use ramp loss function to restrict the maximal loss value of outliers to a fixed value. Therefore, they both are sparse and robust algorithms. They both are concave–convex optimization problems. In each CCCP, Ramp-TKSVC solves a sequence of TKSVC problems while RKWMTSVM solves a series of KWMTSVM problems. Compared to Ramp-TKSVC, RKWMTSVM considers the structural information by KNN which can greatly improve the generalization ability and reduce the redundant constraints.

For a 3-class classification problem, the time computational complexity of TKSVC is $O(2 \times (2l/3)^3)$ and then, computational complexity of Ramp-TKSVC is about $O(l^2\log(l))$ times the iteration number.

4.3.6 LS-KWMTSVM vs. RKWMTSVM

Both LS-KWMTSVM (Tanveer et al 2021a) and the proposed RKWMTSVM are multi-classification algorithms. They both exploit the local information of training samples by KNN. The LS-KWMTSVM is a least squares version of KWMTSVM, because it uses 2-norm instead of 1-norm in KWMTSVM. Therefore, LS-KWMTSVM only needs to solve two systems of linear equations rather than two QPPs in KWMTSVM, making LS-KWMTSVM work faster.

For a 3-class classification problem, LS-KWMTSVM contains two systems of linear equations and KNN steps to compute weight matrices. Its total computational complexity is less than $O(l^2\log(l) + 16l^3/27)$.

5 Numerical experiments

To evaluate the performance of our algorithm, in this section, we conduct the experiments on one artificial dataset and twenty-four benchmark datasets which are collected from UCI machine learning repository [1] and Libsvm library [2]. The characteristics of datasets are shown in Table 1. We remove the missing values in dataset Dermatology and replace the missing values with zero in dataset Autompg before doing the partitions. Furthermore, in order to eliminate the influence of dimension and make the numerical calculation easier, we scale all the features into the range of [0, 1] in the data preprocessing.

In order to make the results more convincing, the respected dataset splits into training sets and independent test sets according to k-fold cross-validation method (Salzberg 1997). This method requires that the dataset be divided into k parts, which each of the k parts acted as an independent holdout test set. Because all k parts will be used in the testing part equally and independently, the reliability of the results could be convincing. The accuracy of the algorithm is the averaged one of k experiments. According to the work of Gutierrez et al (2016); Bamakan et al (2017); Tang et al (2021), we also use fivefold cross-validation in our experiment. All experiments are carried out in Matlab R2014a on Windows 7 running on a PC with system configuration Inter Core i3-4160 Duo CPU (3.60GHz) with 12.00 GB of RAM. We report testing accuracy to evaluate the performance of classifiers. "Accuracy" (Acc) denotes the mean value of five times testing results and plus or minus the standard deviation(std). "Time" denotes the average time of five experiments, and each experiment's time consists of training time and testing time. For the nonlinear case, we only adopt the Gaussian kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-r||\mathbf{x}_i - \mathbf{x}_j||^2)$ as it has been proved it always yields higher testing accuracy.

5.1 Experiment on one artificial dataset

In this subsection, we conduct the experiment on one artificial dataset in $\mathbb{R}^2$ dimensional space, which comes from the work of Bamakan et al (2017). Firstly, we generate a training set $T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots (\mathbf{x}_l, y_l)\}$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $l = 150$. The data have three different classes, i.e., the positive, negative and zero class. Each class has 50 samples and follows a Guassian distribution, where $\mu_A = [4\ 3]$,

[1] http://archive.ics.uci.edu/ml/datasets.html

[2] https://www.csie.ntu.edu.tw/~cjlin/libsvm

Table 1 The characteristics of twenty-four benchmark datasets

Datasets	# Samples	# Features	# Classes	# The distribution of classes	# Sources
Iris	150	4	3	50/50/50	UCI
Soybean	47	35	4	10/10/10/17	UCI
Teaching	151	5	3	49/50/52	UCI
Wine	179	13	3	59/71/49	UCI
Thyroid	215	5	3	150/35/30	UCI
Vertebral	310	7	3	60/150/100	UCI
Glass	214	10	6	70/76/17/13/9/29	UCI
Dermatology	358	34	6	111/60/71/48/48/20	UCI
Svmguide2	391	20	3	221/117/53	Libsvm
Svmguide4	300	10	6	44/56/44/56/47/53	Libsvm
Autompg	398	7	3	249/70/79	UCI
Ecoli	327	7	5	143/77/35/20/52	UCI
Seeds	210	7	3	70/70/70	UCI
Hayes	132	5	3	51/51/30	UCI
Balance	625	4	3	49/288/288	UCI
Waveform	900	21	3	300/300/300	UCI
Vehicle	846	18	4	199/217/218/212	UCI
Zoo	101	17	7	41/20/5/13/4/8/10	UCI
Cmc	1473	10	3	629/333/511	UCI
Car	1728	6	4	1210/384/69/65	UCI
Optidigits	2268	64	4	571/557/552/568	UCI
Yeast	1484	8	10	463/429/244/163/51/44/37/30/20/5	UCI
Satimage	4220	36	4	1533/703/1358/626	UCI
PageBlock	5473	10	5	4913/329/28/88/115	UCI

$\mu_B = [1\ 1]$, $\mu_C = [3\ -3]$, and

$$\Sigma_A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \Sigma_B = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}, \Sigma_C = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}.$$

Similarly, to eliminate the influence of dimension and make the numerical calculation easier, we scale all the features into the range of [0, 1]. The corresponding results are shown in Figs. 4, 5 and 6. The blue "⋆", red "►" and green "⧫" stand for the positive, negative and zero class, respectively. The "∘" and "△" in Fig. 4 stand for the samples selected by KNN in the QPP (16) and QPP (17), respectively. Figure 4a and b shows the results of setting cluster parameter $k = 3$ and $k = 4$, respectively. The results show that the more samples will be removed by setting smaller k. Besides, the appropriate k will help to remove some noises, such as the green point in the lower-left corner which is a outlier of the rest class.

In Figs. 5 and 6, the red and black thick solid line represent the decision boundary $f_+ = -1 - \varepsilon$ and $f_- = 1 + \varepsilon$, respectively. The "∘" and "△" denote support vectors (SVs) in the first and second QPP in both KWMTSVM and RKWMTSVM, respectively. Figure. 5 and 6 show the results of KWMTSVM and RKWMTSVM with different parameter ε and kernel parameter r under cluster parameter $k = 3$ and $k = 4$, respectively. It is obvious that the number of SVs of RKWMTSVM is less than that of KWMTSVM under the same parameter. The most obvious point is the red triangle which locates in the upper right corner which is regarded as a outlier of negative class. Obviously, it is marked by the circle which means it is a SV in the KWMTSVM while it is not a SV any more in our RKWMTSVM. The results show that our RKWMTSVM is more sparse compared with the KWMTSVM. The results also indicate that the number of SVs is different under different parameters. Besides, the positive decision boundary (the red one) obtained by the KWMTSVM is more shifted toward the outlier compared with our RKWMTSVM. The results demonstrate that the proposed RKWMTSVM is more robust compared with the KWMTSVM.

5.2 Influence of parameters

We conduct the experiments on thirteen datasets to investigate the influence of cluster parameter k in our RKWMTSVM. The results are shown in Table 2, where we record the testing accuracy of RKWMTSVM with the cluster parameter

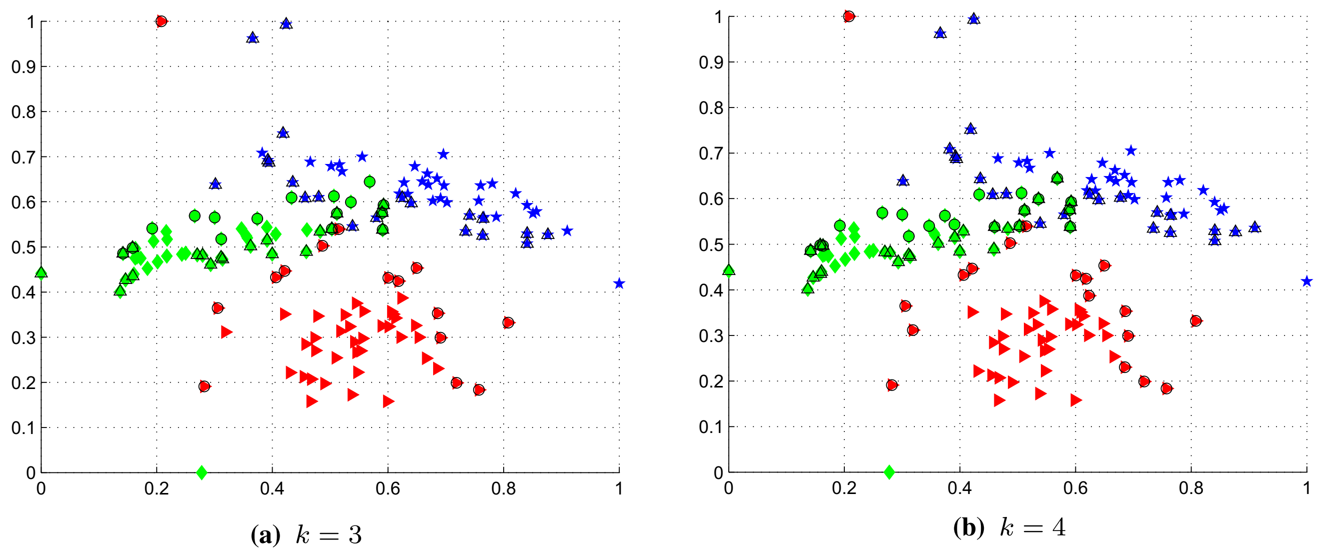


Fig. 4 The samples selected by KNN, where "∘" stands for the samples with $f_j = 1, j \in I_2$ and $h_k = 1, k \in I_3$, "△" stands for the samples with $f_i = 1, i \in I_1$ and $h_k = 1, k \in I_3$

k ranges from 3 to 8. As the increase of k, the computational time of the proposed RKWMTSVM tends to rise. A smaller k will help RKWMTSVM to remove the redundant data and reduce the computational time. When $k = max\{l_1 + l_3, l_2 + l_3\}$, no training samples will be removed by KNN and the computational time will become longer. With the increase of k, the change of classification accuracy is not obvious.

In order to analyze the influence of parameter c_2 and kernel parameter r on the predictive accuracy in the KWMTSVM and our RKWMTSVM, we do the experiments on four benchmark datasets, i.e., Autompg, Glass, Teaching and Wine, and the results are shown in Figs. 8, 9, 10, 11, where we search the parameter c_2 and r from the range $\{2^i|i = -4, ..., 6\}$ and fix $c_1 = 4, \varepsilon = 0.2, k = 3$. From Fig. 8, it is hard to judge which parameter affects more on the testing accuracy. In Figs. 8 and 9, it appears that the impact of kernel parameter r is more than penalty parameter c_2, and both KWMTSVM and RKWMTSVM achieve better testing accuracy for smaller value of parameter r. However, it seems that the classifiers achieve better predictive accuracy for large value of r in Fig. 10.

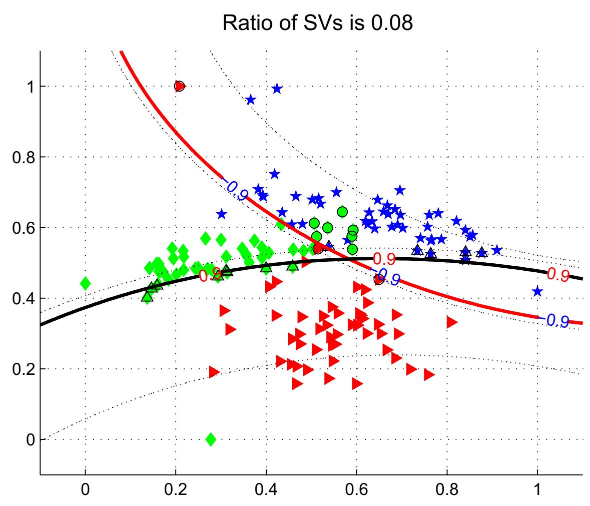
Because the proposed RKWMTSVM contains too many parameters, we carry out the analysis of variance (ANOVA) (Hamdia et al 2018) decomposition to do the key parameter sensitivity analysis. ANOVA is used to quantify the independent and joint effects of the parameters on 5-fold cross validation accuracy and calculate the sensitivity index of each parameter (S_{T_i}). The works of Hamdia et al (2018) and Long et al (2020) show the specific calculation, and they conclude that the larger S_{T_i}, the higher sensitivity of accuracy with respect to the i-th parameter. The results using ANOVA on our RKWMTSVM are shown in Fig. 7, where the x-axis denotes six benchmark datasets and y-axis is the indices S_T. The parameter r has a dramatic effect on accuracy, followed by c_1 and c_2. Relatively, the sensitivity brought by ε and k is much weaker. Summarily, the parameter k can help to remove the redundant constraints and improve the classification accuracy of RKWMTSVM, but their variation cannot lead to strong accuracy fluctuation.

Based on the results discussed above, we set $k = 3$ in the following experiments to reduce the long searching time.

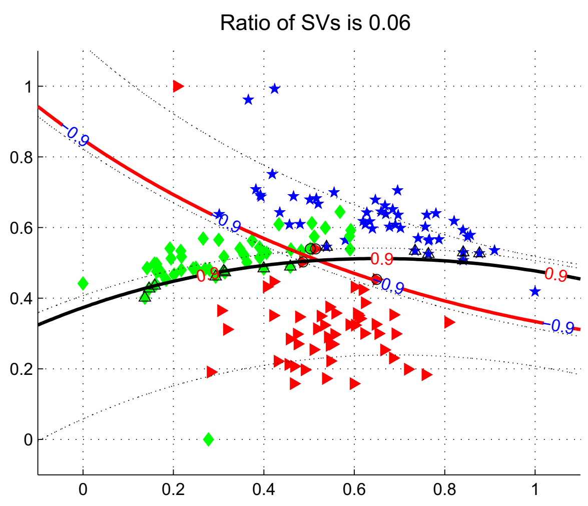
5.3 Parameter selection

An important preliminary issue for SVMs is the optimal parameter selection. There are three types of parameter optimization methods of SVMs. The first is the typical of grid search method (Friedrichs and Igel 2005). The second is numerical optimization method, such as gradient descent method (Schölkopf et al 2007). Many meta-heuristics methods are also introduced to obtain an acceptable optimum, including genetic algorithms, particle swarm optimization, bat algorithms, etc. (Wang et al 2013; Tharwat et al 2017; Li et al 2018). In this paper, we adopt the traditional grid search method to get the optimal parameters.

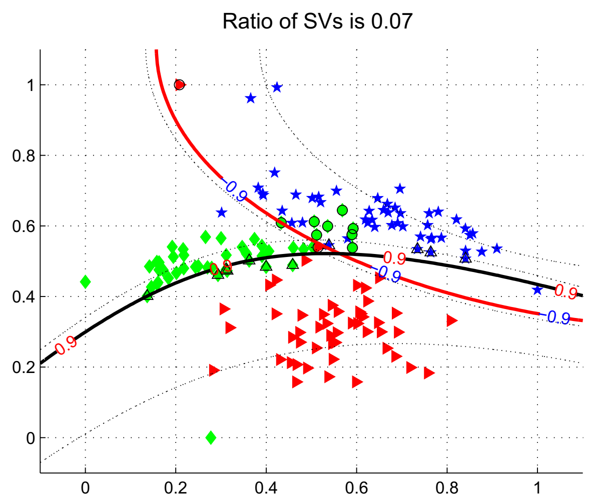
The kernel parameter r is searched from $\{2^i|i = -4, ..., 6\}$. In order to reduce the computational complexity of parameter selection, we set $c_1 = c_2$ in OVO PGTSVM, OVR WTSVM and OVR RνTSVM; $\tau_1 = \tau_2$ in OVO PGTSVM; $\nu_1 = \nu_2$ and $\sigma_1 = \sigma_2$ in OVR Rν-TSVM; $c_1 = c_3$, $c_2 = c_4$ in Ramp-TKSVC, KWMTSVM, LS-KWMTSVM and RKWMTSVM. In general, we need to choose (c_1, τ_1, r) in OVO PGTSVM; (c_1, r) in OVR WTSVM; (ν_1, σ_1, r) in OVR



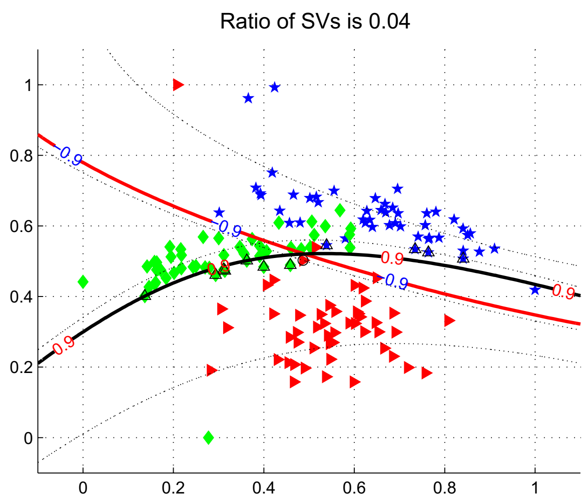
(a) KWMTSVM, $\varepsilon = 0.1, r = 0.1$



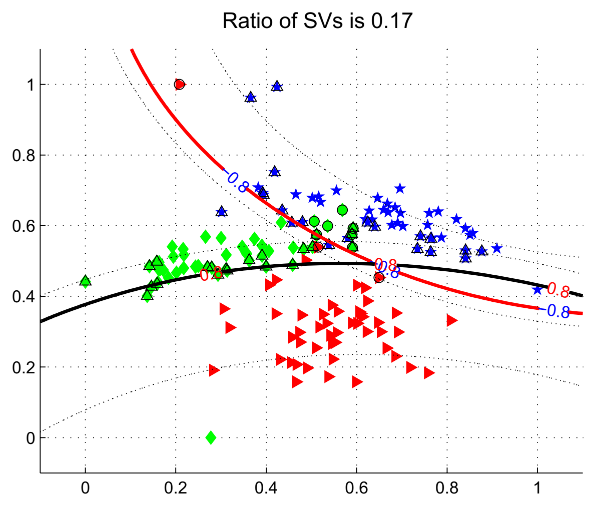
(b) RKWMTSVM, $\varepsilon = 0.1, r = 0.1$



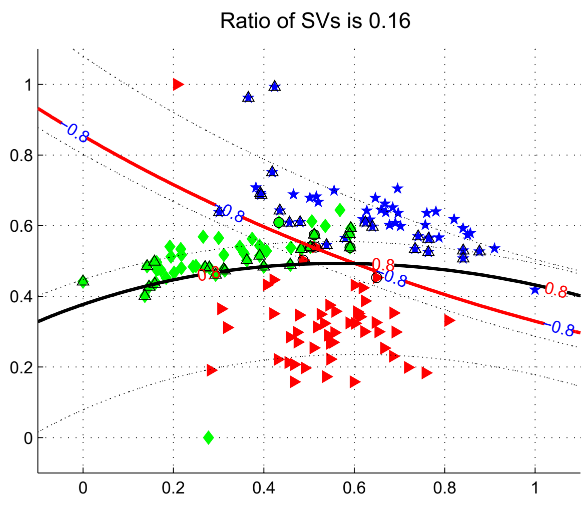
(c) KWMTSVM, $\varepsilon = 0.1, r = 0.2$



(d) RKWMTSVM, $\varepsilon = 0.1, r = 0.2$

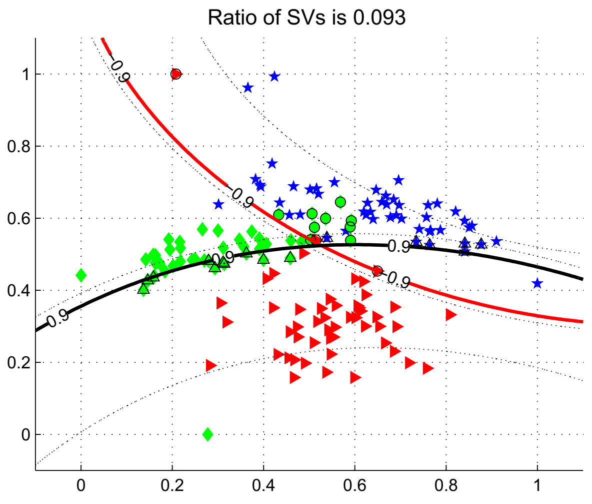


(e) KWMTSVM, $\varepsilon = 0.2, r = 0.1$

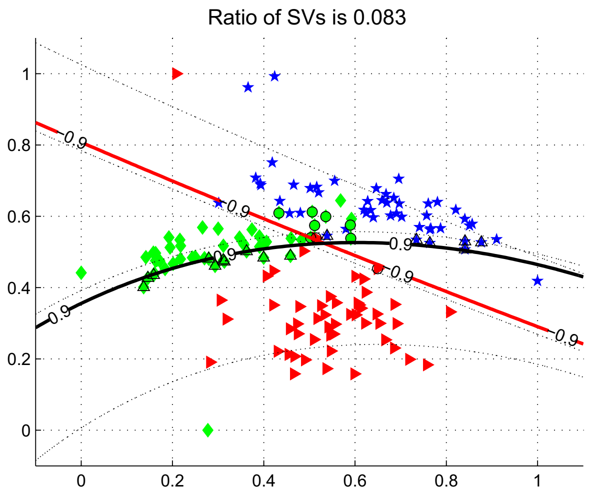


(f) RKWMTSVM, $\varepsilon = 0.2, r = 0.1$

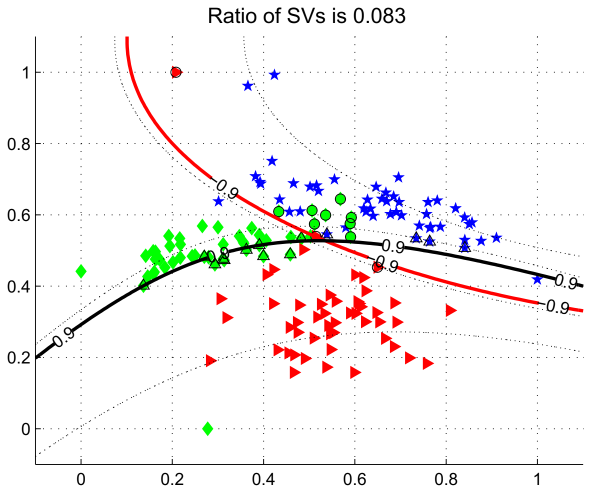
Fig. 5 The plots of KWMTSVM and RKWMTSVM with different parameters with cluster parameter $k = 3$, where $c_i = 5$, "$\circ$" and "$\triangle$" stand for the SVs



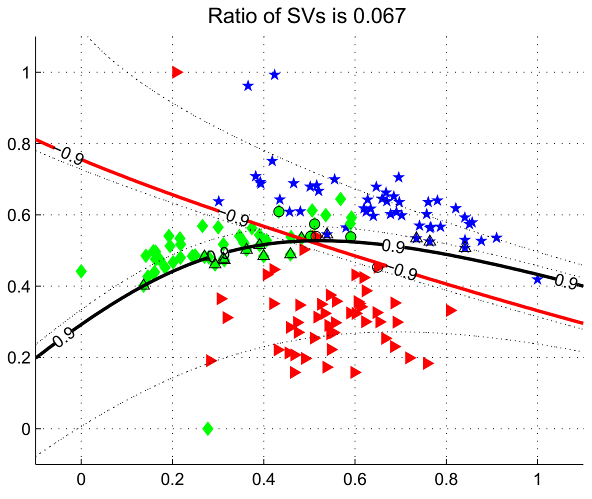
(a) KWMTSVM, $\varepsilon = 0.1, r = 0.1$



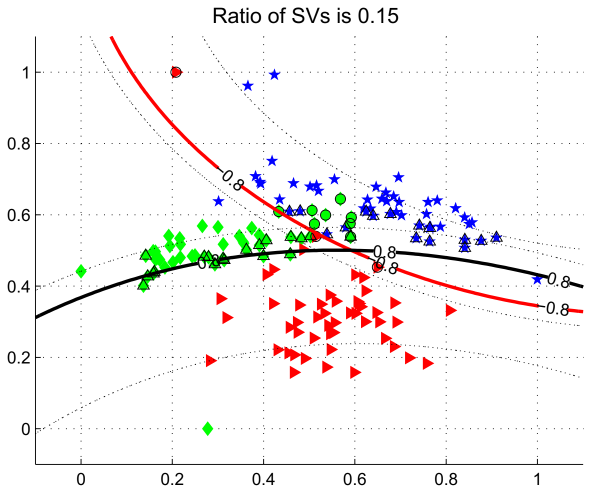
(b) RKWMTSVM, $\varepsilon = 0.1, r = 0.1$



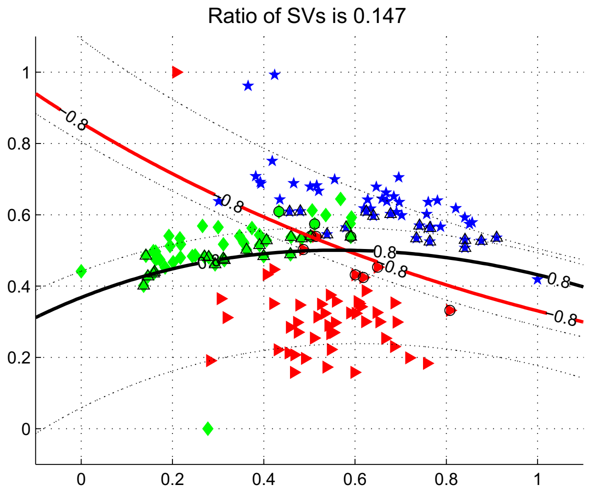
(c) KWMTSVM, $\varepsilon = 0.1, r = 0.2$



(d) RKWMTSVM, $\varepsilon = 0.1, r = 0.2$



(e) KWMTSVM, $\varepsilon = 0.2, r = 0.1$



(f) RKWMTSVM, $\varepsilon = 0.2, r = 0.1$

Fig. 6 The plots of KWMTSVM and RKWMTSVM with different parameters with cluster parameter $k = 4$, where $c_i = 5$

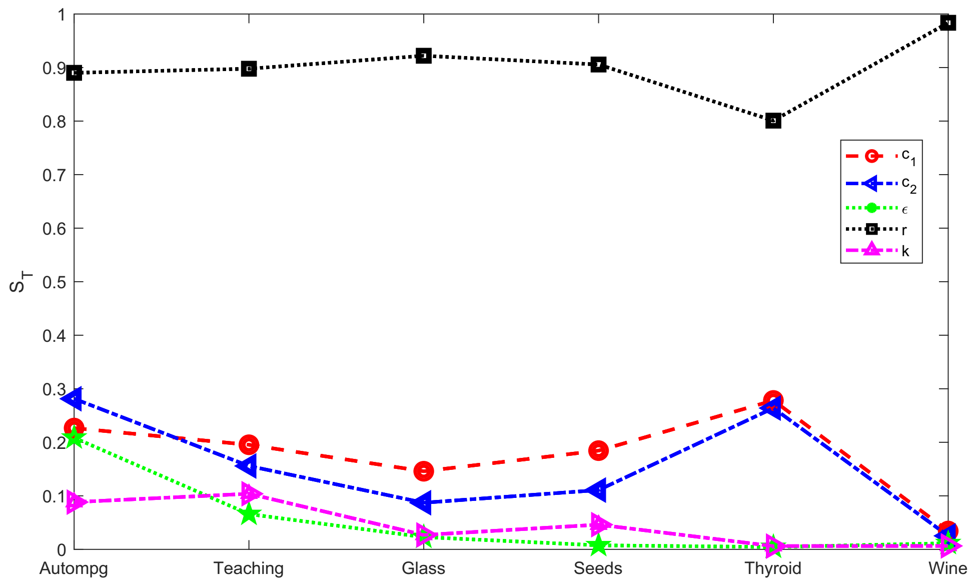
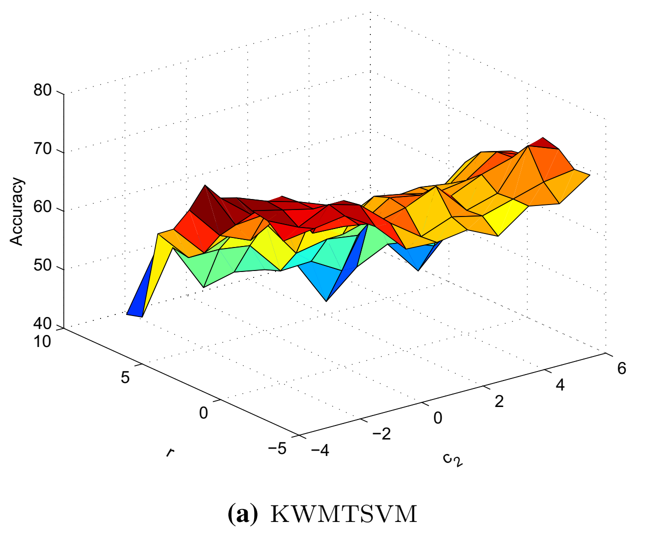


Fig. 7 S_T indices of the parameters

Table 2 Performance comparisons of RWMTSVM with different cluster parameter k on thirteen datasets

Datasets	$k = 3$ Acc(%)±std Time(s)	$k = 4$ Acc(%)±std Time(s)	$k = 5$ Acc(%)±std Time(s)	$k = 6$ Acc(%)±std Time(s)	$k = 7$ Acc(%)±std Time(s)	$k = 8$ Acc(%)±std Time(s)
Iris	98.67±2.98	98.00±4.47	98.00±4.47	98.67±2.98	98.67±2.98	98.00±4.47
	0.1480	0.1244	0.1046	0.1745	0.2103	0.1970
Balance	93.17±4.07	93.33±3.71	93.97±3.75	93.81±4.02	93.81±3.48	93.97±3.58
	3.0316	1.5345	2.4639	3.2054	3.0748	3.2618
Teaching	67.74±12.28	66.45±4.89	68.39±22.16	67.74±10.45	67.74±10.20	67.74±11.40
	0.1491	0.2486	0.2232	0.2572	0.3183	0.3319
Thyroid	97.21±5.04	97.21±5.04	97.21±5.04	97.21±5.04	97.21±5.04	97.21±5.04
	0.2090	0.2154	0.2185	0.2187	0.2633	0.2252
Glass	94.17±6.32	92.92±10.37	93.75±6.07	93.75±4.42	93.33±7.13	92.50±4.56
	1.1727	0.7883	0.7932	1.2285	2.5534	2.2137
Dermatology	96.22±2.93	95.68±3.75	96.22±2.60	96.49±2.63	97.57±1.48	97.57±1.48
	1.5705	1.1609	1.4599	2.0033	1.5689	1.3668
Autompg	75.50±5.20	76.25±5.99	76.75±4.97	76.75±5.27	75.50±5.63	74.75±6.02
	0.8053	0.7167	0.8678	0.9585	0.9130	0.7680
Seeds	94.76±9.13	94.29±8.00	94.76±6.61	93.81±8.68	93.81±7.26	93.81±8.68
	0.1672	0.1656	0.1810	0.1734	0.1818	0.1967
Cmc	98.18±0.91	98.18±0.78	98.18±0.91	98.24±0.65	98.38±0.65	98.24±0.65
	13.4340	15.6712	15.4758	18.2348	19.0456	20.8493
Optidigits	99.65±0.50	98.55±1.19	98.90±0.88	98.99±0.55	99.04±0.55	99.04±0.55
	34.9574	29.6381	29.4859	29.5612	29.4007	30.4249
Hayes	85.00±5.30	84.29±4.07	84.29±5.42	83.57±5.42	85.00±5.87	85.71±5.65
	0.1865	0.2243	0.2177	0.2277	0.2548	0.2711
Zoo	98.18±4.07	98.18±4.07	98.18±4.07	98.18±4.07	98.18±4.07	98.18±4.07
	1.1785	0.9893	1.0626	1.2154	1.0800	1.0196
Svmguide4	76.83±1.81	74.92±5.19	74.92±6.29	74.60±3.17	75.87±1.33	76.51±2.35
	2.3470	3.4243	4.0595	2.2887	2.6648	5.5291



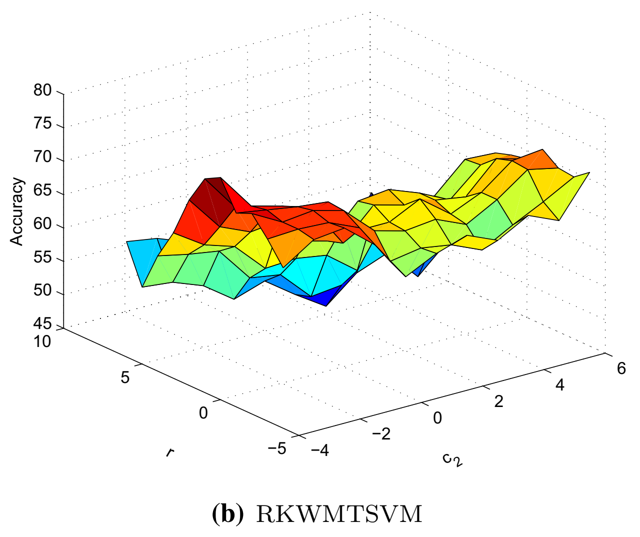
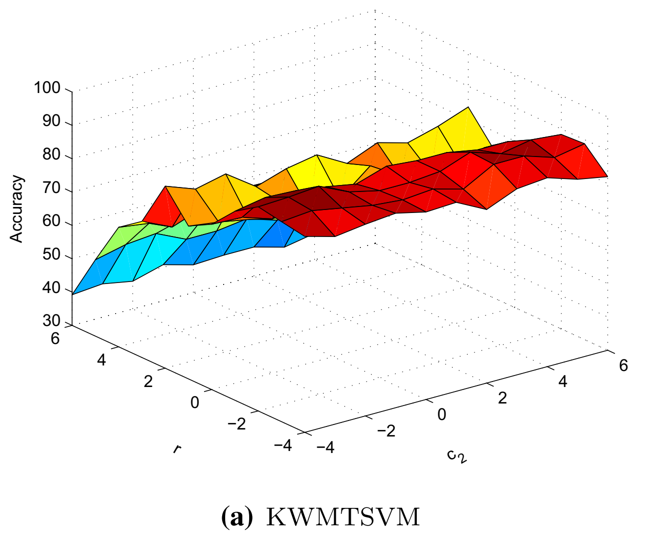


Fig. 8 The relationship of kernel parameter r, parameter c_2, and classification accuracy in the KWMTSVM and RKWMTSVM on Autompg dataset, respectively



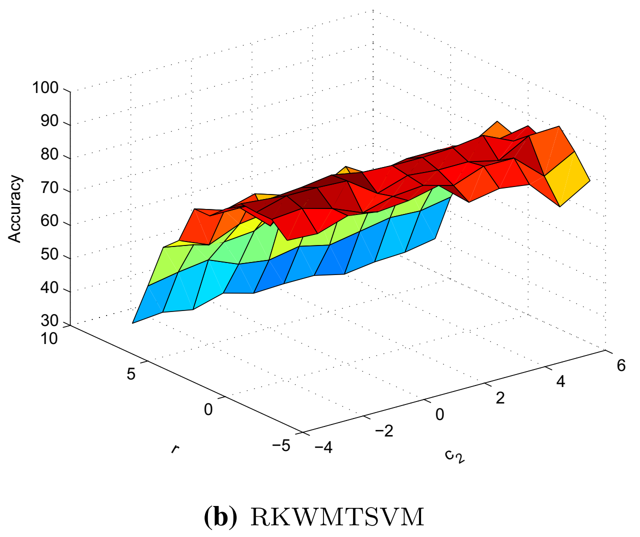


Fig. 9 The relationship of kernel parameter r, parameter c_2, and classification accuracy in the KWMTSVM and RKWMTSVM on Glass dataset, respectively

Rν-TSVM; $(c_1, c_2, \varepsilon, r)$ in Ramp-TKSVC, KWMTSVM, LS-KWMTSVM and our RKWMTSVM. The parameter c_1 is selected from the set $\{2^i|i = -4, ..., 6\}$. The parameter ν_1 is searched from the set $\{0.2, ..., 0.7\}$. The parameter τ_1 is searched from the set $\{0.1, 0.5, 0.9\}$. The value σ_1 is searched from the set $\{1.5, 2, 2.5, 3, 5, 8\}$. The parameter ε ranges from set $\{0.1, 0.2, 0.3\}$. For large-scaled datasets, we reduce the number of parameters uniformly by enlarging the search step due to the long running time.

5.4 Result comparisons and discussions

Table 3 reports the averages and standard deviations of the testing accuracies (in %) and running times (s) for the seven compared algorithms on twenty-four datasets. The bold value indicates the best predictive accuracy.

From the perspective of testing accuracy, we can learn that our proposed RKWMTSVM outperforms the other six algorithms for most datasets. The reasons are twofold. On one hand, our RKWMTSVM retains the property of the KWMTSVM which utilizes the local information of inter-class and intra-class by using KNN method. On the other hand, the introduction of ramp loss makes our RKWMTSVM insensitive to outliers. Besides, we find that OVO PGTSVM performs better than OVR WTSVM and OVR RνTSVM for most cases. A possible reason is that OVR strategy easily leads to imbalance problem which may lead to bad performance. Another possible reason is that OVO

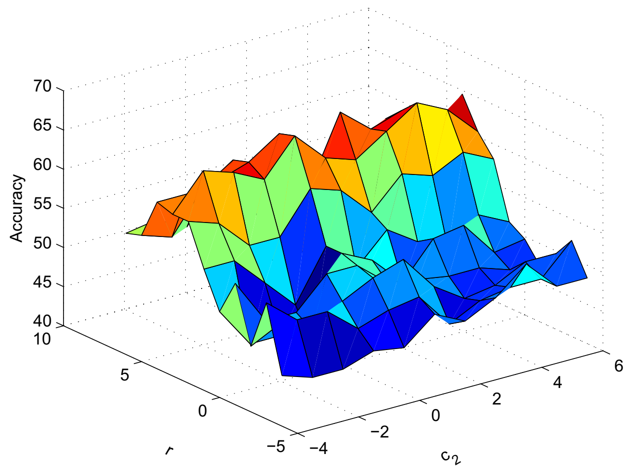
Table 3 Performance comparisons of seven nonlinear algorithms with Gaussian kernel

Datasets	OVO PGTSVM Acc(%)±std Time(s)	OVR WTSVM Acc(%)±std Time(s)	OVR RνTSVM Acc(%)±std Time(s)	KWMTSVM Acc(%)±std Time(s)	Ramp-TKSVC Acc(%)±std Time(s)	LS-KWMTSVM Acc(%)±std Time(s)	RKWMTSVM Acc(%)±std Time(s)
Iris	**98.67±1.83**	98.00±2.98	98.00±2.98	**98.67±2.98**	98.00±2.98	**98.67±2.98**	**98.67±2.98**
	0.0682	0.0942	0.1207	0.1158	0.0445	0.0230	0.1480
Soybean	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**
	0.0537	0.0546	0.0403	0.0810	0.1078	0.0581	0.0836
Teaching	64.52±26.31	65.16±5.30	67.10±22.63	66.45±4.89	**72.00±25.56**	67.74±6.45	67.74±12.28
	0.0514	0.1148	0.0992	0.1329	0.0729	0.0213	0.1491
Wine	**97.84±2.26**	96.22±4.10	**97.84±2.26**	96.76±4.44	**97.84±2.26**	97.30±3.82	**97.84±3.52**
	0.9449	0.1212	0.0722	0.0908	0.3715	0.0930	0.1083
Thyroid	95.81±3.82	**97.21±3.82**	**97.21±5.04**	**97.21±5.04**	95.35±5.70	96.74±5.10	**97.21±5.04**
	0.1045	0.1953	0.1345	0.1430	0.1063	0.0769	0.2090
Vertebral	79.03±32.82	82.90±4.78	83.23±5.77	84.52±7.53	84.52±34.78	84.19±8.02	**85.16±4.89**
	0.1621	0.4565	0.2248	0.2875	0.2570	0.3222	0.5026
Glass	92.50±3.49	87.92±5.39	87.50±7.93	93.33±6.97	**96.44±3.72**	88.33±9.50	94.17±6.32
	0.5573	0.5084	0.4217	1.0065	0.3163	0.4518	1.1727
Dermatology	**98.11±0.74**	94.59±2.87	94.59±3.45	95.68±1.13	97.30±1.66	92.97±3.63	96.22±2.93
	0.7205	1.1121	1.0510	0.8638	1.9331	0.4211	1.5705
Svmguide2	82.75±4.37	79.75±5.41	79.25±3.01	82.50±4.24	83.25±2.88	82.75±2.71	**83.50±3.89**
	0.2718	0.7846	0.3989	0.5508	0.7017	0.4134	0.7290
Svmguide4	71.75±7.13	57.46±4.11	57.46±13.81	75.24±6.21	76.51±5.54	71.75±2.07	**76.83±1.81**
	0.4047	0.9652	0.6493	1.3062	3.7802	0.1528	2.3470
Autompg	71.75±6.03	70.25±4.28	72.75±9.94	76.00±3.99	75.75±5.12	**77.50±2.34**	75.50±5.20
	0.4751	0.6759	0.4443	0.4724	4.3330	0.4783	0.8053
Ecoli	83.58±5.28	85.07±7.08	86.87±7.78	**88.36±4.99**	87.46±3.44	87.46±4.55	**88.36±3.06**
	0.6460	0.7716	0.6490	0.9149	0.5800	0.5837	1.2654
Seeds	**96.67±3.98**	92.38±9.13	92.38±10.43	94.29±8.84	91.43±9.00	94.76±9.13	94.76±9.13
	0.0890	0.1890	0.1248	0.1191	0.1162	0.1165	0.1672

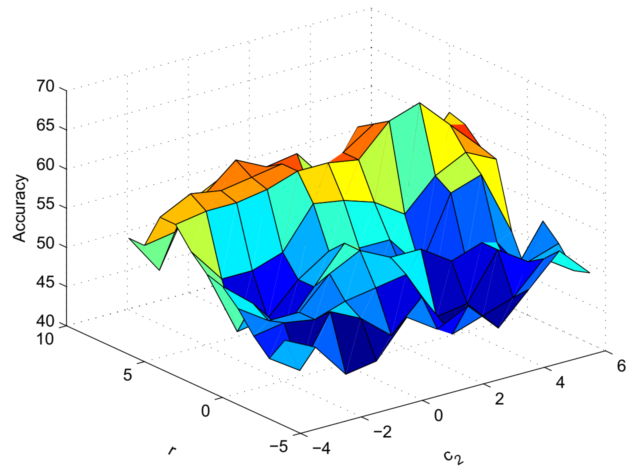
Table 3 continued

Datasets	OVO PGTSVM Acc(%)±std Time(s)	OVR WTSVM Acc(%)±std Time(s)	OVR RνTSVM Acc(%)±std Time(s)	KWMTSVM Acc(%)±std Time(s)	Ramp-TKSVC Acc(%)±std Time(s)	LS-KWMTSVM Acc(%)±std Time(s)	RKWMTSVM Acc(%)±std Time(s)
Hayes	82.14±10.41	80.00±10.29	80.71±9.31	83.57±11.18	80.71±6.49	**86.43±8.53**	85.00±5.30
	0.1342	0.0894	0.0735	0.0823	0.2506	0.0709	0.1865
Balance	89.84±6.63	90.16±8.90	91.27±7.55	93.02±4.40	90.72±1.21	78.25±10.14	**93.17±4.07**
	3.5058	1.4736	4.1513	2.0982	1.0603	0.5624	3.0316
Waveform	86.11±3.02	85.56±2.64	86.44±2.53	85.11±1.77	**86.67±1.96**	86.56±2.27	86.22±1.77
	7.3307	4.4767	1.9601	5.1298	19.0180	1.4002	5.8813
Vehicle	**85.61±0.89**	81.17±3.32	81.40±1.72	85.03±2.21	84.33±3.82	81.40±3.13	84.33±1.51
	0.6907	5.6526	3.4836	3.5941	60.9884	0.7738	11.6817
Zoo	**100.00±0.00**	99.09±2.03	98.18±2.49	98.18±4.07	98.18±43.45	98.18±4.07	98.18±4.07
	0.5236	0.6173	0.5866	1.2609	0.9645	0.2537	1.1785
Cmc	**99.86±0.30**	96.96±1.37	96.96±1.51	98.18±0.91	99.53±0.39	98.38±1.00	98.18±0.91
	3.0950	12.6029	7.8421	11.2018	56.7435	11.3591	13.4340
Car	92.72±3.23	81.62±2.70	85.95±2.21	91.62±4.18	**93.06±5.80**	88.32±1.89	93.01±3.40
	81.8868	47.7279	51.4678	35.4854	229.0359	24.8015	37.7092
Optidigits	**99.91±0.12**	99.65±0.29	99.39±0.29	99.65±0.50	99.21±0.57	98.07±0.73	99.65±0.50
	16.1935	45.9626	65.2784	35.1810	234.0370	18.8997	34.9574
Yeast	51.91±1.93	50.90±5.22	52.24±0.69	57.86±2.08	51.37±2.51	55.72±2.42	**59.06±1.93**
	85.5187	134.5751	109.5475	54.2197	764.1713	36.7500	113.8112
Satimage	**93.78±2.44**	92.22±0.88	93.26±3.16	92.25±3.70	93.19±3.68	88.79±4.82	92.84±4.04
	414.4826	406.0727	461.2994	349.8086	2119.6430	205.3121	555.1703
Pageblock	92.54±2.57	**95.18±1.79**	93.91±1.80	94.00±1.05	94.56±0.96	94.36±1.09	95.07±1.73
	6212.8288	1592.1354	1030.2899	745.1607	5799.2300	402.3769	1070.8629
Average rank	3.83	5.52	4.94	3.65	3.52	4.19	2.54

The bold values indicate the best predictive accuracies

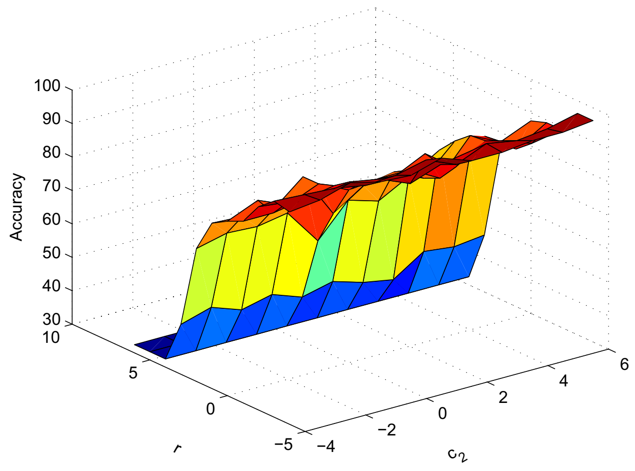


(a) KWMTSVM

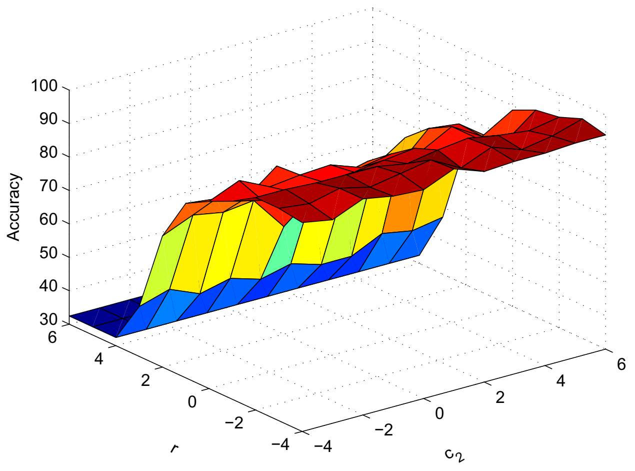


(b) RKWMTSVM

Fig. 10 The relationship of kernel parameter r, parameter c_2, and classification accuracy in the KWMTSVM and RKWMTSVM on Teaching dataset, respectively



(a) KWMTSVM



(b) RKWMTSVM

Fig. 11 The relationship of kernel parameter r, parameter c_2, and classification accuracy in the KWMTSVM and RKWMTSVM on Wine dataset, respectively

PGTSVM adopts the pinball loss function which maximize the quantile distance rather than the shortest distance between classes, making the algorithm be less sensitive to noises. OVR RνTSVM performs better than OVR WTSVM on most datasets. The reason for that is RνTSVM punishes the misclassified samples according to their positions which can improve the performance. The algorithms which take one-verse-one-verse-rest structure, i.e., KWMTSVM, Ramp-TKSVC, LS-KWMTSVM and our RKWMTSVM, perform better than binary classifiers adopted OVR strategies, i.e., OVR WTSVM and OVR RνTSVM. The reason is that KWMTSVM, Ramp-TKSVC, LS-KWMTSVM and our RKWMTSVM can not only utilize all the information of training data points in every classifier but also avoid the imbalance problem to some extent. Ramp-TKSVC performs best on six datasets, while KWMTSVM and LS-KWMTSVM perform best on four datasets, which implies that Ramp-TKSVC yields better. That is because Ramp-TKSVC also adopts ramp loss function to avoid the disturbance of outliers. The seven algorithms all yield 100% testing accuracy on Soybean dataset, which implies that seven algorithms yield the comparable performance.

From the view of running time, when the size of dataset is relatively small, it is hard to judge which algorithm has the least running time. The three strategies, i.e., OVO, OVR, and OVOVR, have their pros and cons. When the size of dataset is large scale, the advantage of KWMTSVM, LS-KWMTSVM and our RKWMTSVM is obvious. That is because they all

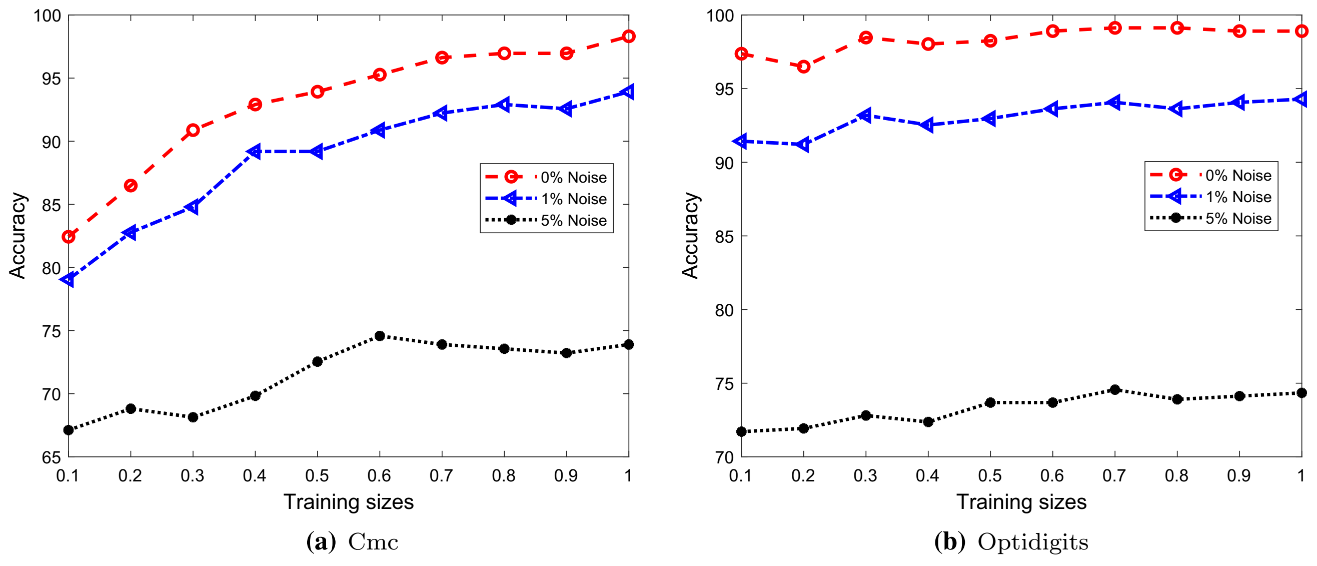


Fig. 12 The accuracy variation with different training sizes

can remove the redundant constraints by KNN, especially when they are compared with Ramp-TKSVC. It is worth to note that LS-KWMTSVM takes the least time among the seven algorithms. Because LS-KWMTSVM solves linear equations and uses KNN to remove the consonant constraint. Although both Ramp-TKSVC and RKWMTSVM adopt ramp loss function to depress the negative influence of outliers, the proposed RKWMTSVM works faster than Ramp-TKSVC.

5.5 Training efficiency

To see the performance of the proposed RKWMTSVM on different training sizes of noisy datasets, we do experiments on Cmc and Optidigits datasets which have larger sample sizes. The results are shown in Fig. 12, where x-axis denotes the training sizes and y-axis denotes the testing accuracy. The red, blue and black lines are the case with no noise, 1% and 5% label noise added, respectively. The results show that the accuracy of RKWMTSVM will increase with the increase in training sizes. Fig. 13 is the time change with different training sizes, where y-axis is the logarithmic of time. Obviously, the training time will increase with the training size increases.

In addition, we discuss the training time of RKWMTSVM, which needs to solve optimization problems (36) and (37) repeatedly by CCCP. Suppose that each outer loop requires a comparable amount of running time, leading to a sharp increase in the total training time with the number of iterations. To verify the hypothesis, we draw Fig. 14 to reveal how the training time distributes during the outer loop process on five datasets, where t_i denotes the running time of ith subproblem. Figure 14 denotes the curve of t_i/t_1 ratio. It

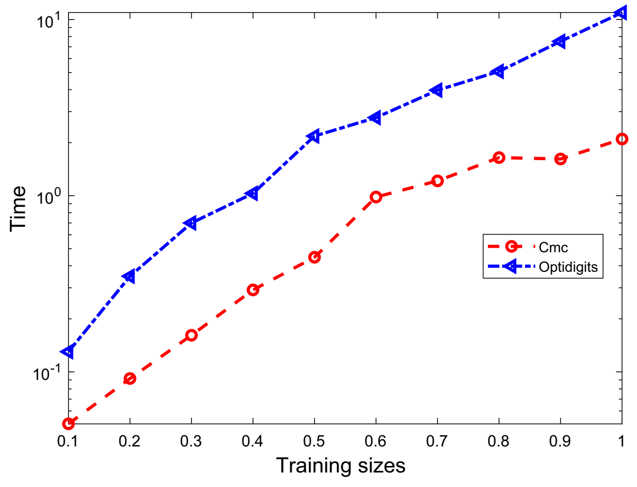


Fig. 13 The time variation with different training sizes

displays that the five datasets will be terminated less than 8 iterations. Datasets Wine and Thyroid take only 3 iterations. In addition, after the first iteration, the running time of the following iterations decreases drastically. It is obvious that Iris dataset takes 8 iterations to converge while the first iteration takes the most training time. These results prove the effectiveness of the proposed Algorithm 1.

5.6 Statistical tests

From Table 3, we can see that our RKWMTSVM not always performs the best among the seven algorithms on the twenty-four datasets from the perspective of testing accuracy. To further analyze the performance of seven algorithms on multiple datasets statistically, we use Friedman test (Demsar

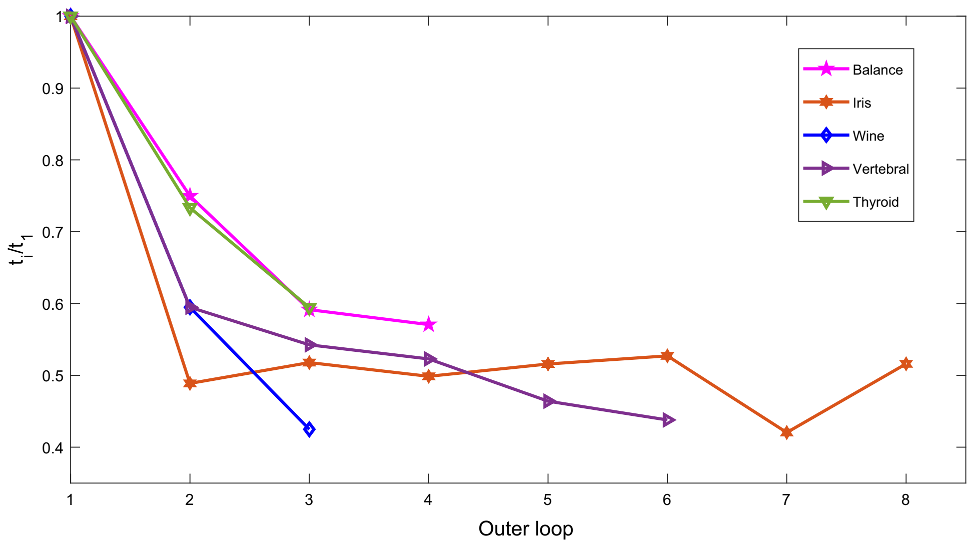


Fig. 14 The variations of t_i/t_1 as the outer loop iterations on five datasets

2006) and Holm–Bonferroni test (Holm 1979) which are two commonly used statistical methods.

5.6.1 Friedman test

The Friedman test is suggested by Demsar (2006) and Salvador et al (2010), and it is proved to be simple, nonparametric and safe for comparing three or more related samples. For this, the average ranks of seven algorithms on accuracy for twenty-four datasets are listed in the last line of Table 3.

Referring to the works of Wang et al (2020); Tanveer et al (2021a), we still suppose the null-hypothesis be that all the algorithms are equivalent. We can obtain the Friedman statistic according to the following equation,

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right], \tag{44}$$

where $R_j = \frac{1}{N}\sum_i r_i^j$, and r_i^j denotes the jth of k algorithms on the ith of N datasets. Friedman's χ_F^2 is undesirably conservative and derives a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}, \tag{45}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

On the basis of (44) and (45), we derive $\chi_F^2 = 37.3551$ and $F_F = 8.0563$, where F_F is distributed according to F-distribution with (6, 138) degrees of freedom. The critical value of $F(6, 138)$ is 1.817 for the level of significance $\alpha = 0.1$; similarly, it is 2.165 for $\alpha = 0.05$. Because the value of F_F is 8.0563 which is much larger than the critical value, it indicates that there is significant difference between the seven algorithms. It is worth to mention that the average rank of the proposed RKWMTSVM is far lower than the remaining algorithms, which means that it has a better performance than the other six compared algorithms.

5.6.2 Holm–Bonferroni test

Holm–Bonferroni method (Simes 1986) is a frequently used method to compare the performance of multiple algorithms. The p value (Demsar 2006) is calculated by performing pairwise t-test, and the proposed RKWMTSVM is statistically compared with the other six algorithms. According to the statement of Demsar (2006), the null hypothesis assumes that the data of two-sample t-test come from independent random samples with equal means and equal but unknown variances. For a given significance level $\alpha = 0.05$, Holm–Bonferroni test orders the p-values from minimum to maximum as $p_{(1)}, p_{(2)}, \ldots, p_{(6)}$ with corresponding null hypothesis $H_{(1)}, H_{(2)}, \ldots, H_{(6)}$. The results of p-values are presented in Table 4 ($\alpha = 0.05$). It rejects the null hypotheses $H_{(1)}, \ldots, H_{(i-1)}$ and does not reject $H_{(i)}, \ldots, H_{(6)}$, if

$$p_{(i)} > \frac{\alpha}{6+1-i}, 1 < i \le 6, \tag{46}$$

where i is the minimal index.

For dataset Iris, $(p_{(2)} = 0.7238 = p_{(3)} = p_{(5)}) < (p_{(1)} = p_{(4)} = p_{(6)} = 1)$ and $p_{(2)} > \alpha/5$. Hence, we conclude that $H_{(1)}, \ldots, H_{(6)}$ are not rejected. That means the RKWMTSVM is statistically similar to OVO PGTSVM, OVR WTSVM, OVR RνTSVM, KWMTSVM, Ramp-TK SVC and LS-KWMTSVM on Iris dataset. In this way, we find 16 out of the 144 null hypotheses are judged which indicates our method has significant advantage over others. In total, the statistical results on p value suggest that the pro-

Table 4 The p values of the seven algorithms using Gaussian kernel

Datasets	OVO PGTSVM	OVR WTSVM	OVR RνTSVM	KWMTSVM	Ramp-TKSVC	LS-KWMTSVM	RKWMTSVM
Iris	1.0000	0.7328	0.7328	1.0000	0.7328	1.0000	–
Soybean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	–
Teaching	0.8125	0.6828	0.9571	0.8355	0.7490	1.0000	–
Wine	1.0000	0.5216	1.0000	0.6816	1.0000	0.8220	–
Thyroid	0.6361	1.0000	1.0000	1.0000	0.5996	0.8882	–
Vertebral	0.1562	0.4817	0.5834	0.8770	0.8536	0.8247	–
Glass	0.6234	0.1319	0.1817	0.8480	0.5114	0.2908	–
Dermatology	0.2264	0.4022	0.4464	0.7156	0.4982	0.1600	–
Svmguide2	0.7817	0.2470	0.0919	0.7078	0.9112	0.7339	–
Svmguide4	0.1896	0.0001	0.0343	0.6083	0.9080	0.0034	–
Autompg	0.3238	0.1207	0.6032	0.8691	0.9408	0.4649	–
Ecoli	0.1273	0.3814	0.7057	1.0000	0.6750	0.7258	–
Seeds	0.6852	0.6909	0.7111	0.9353	0.5770	1.0000	–
Hayes-roth	0.6044	0.3714	0.4038	0.8053	0.2869	0.7600	–
Balance	0.3714	0.5183	0.6368	0.9542	0.2561	0.0266	–
Waveform	0.9459	0.6554	0.8777	0.3566	0.7200	0.8045	–
Vehicle	0.1487	0.1045	0.0219	0.5764	1.0000	0.1112	–
Zoo	0.3739	0.6707	1.0000	1.0000	1.0000	1.0000	–
Cmc	0.0117	0.1432	0.1700	1.0000	0.0259	0.7460	–
Car	0.8938	0.0005	0.0062	0.5814	0.9852	0.0345	–
Optidigits	0.3142	1.0000	0.3477	1.0000	0.2351	0.0053	–
Yeast	0.0004	0.0215	0.0007	0.3701	0.0008	0.0435	–
Satimage	0.6684	0.7547	0.8577	0.8155	0.8882	0.1896	–
Pageblock	0.1099	0.9243	0.3266	0.2758	0.5843	0.4641	–

posed algorithm is not obvious statistically difference with other six algorithms in performance.

In our experiment, the Friedman test reports a significant difference, but the Holm–Bonferroni test which is a post hoc test fails to detect it. This is due to the lower power of the latter (Demsar 2006). In this case, we can only conclude that these algorithms do differ. For example, all seven algorithms yield 100% accuracy on Soybean dataset, and the p values show that they are statistically similar which is rational. The p value of OVR WTSVM is 0.7328 on Iris dataset which indicates that our RKWMTSVM is statistically similar to OVR WTSVM. Actually, the performance of our RKWMTSVM is indeed higher than OVR WTSVM. Another possible reason is that the datasets do not have much outliers, and the advantages of RKWMTSVM are not obvious, especially compared with the robust algorithms OVO PGTSSVM and Ramp-TKSVC.

6 The clipDCD for RKWMTSVM

In this section, we employ the clipDCD algorithm (Peng et al 2014) to accelerate the solving speed of our RKWMTSVM in the training process.

6.1 The clipDCD algorithm

The clipDCD algorithm is a kind of the coordinate descent method, and it is proposed for solving the dual problem of SVM (Peng et al 2014). Till now, it has been successfully embedded into SVM, TSVM, L1-loss-based TSVM (Peng et al 2016), TPMSVM (Peng 2011b), structural TSVM(Pan et al 2015) and so on. In each iteration, the clipDCD algorithm only solves one single-variable sub-problem according to the maximal possibility-decrease strategy on objective value. The dual form of classical SVM is as follows,

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{Q}\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ &s.t. \quad \mathbf{0} \le \boldsymbol{\alpha} \le \mathbf{C}. \end{aligned} \tag{47}$$

It is assumed that only one component of $\boldsymbol{\alpha}$ is updated at each iteration, denoted $\alpha_L = \alpha_L + \lambda$, $L \in \{1, .., n\}$ is the index. It has been proved that

$$f(\lambda) = f(0) + \frac{(e_L - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,L})^2}{2Q_{LL}}, \tag{48}$$

where $\boldsymbol{Q}_{.,L}$ is the Lth column of $\boldsymbol{Q}$. The objective decrease will be approximately largest by maximizing $(e_L - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,L})^2 / Q_{LL}$, and the L index is chosen as

$$L = \arg\max_{i \in \mathcal{A}} \left\{ \frac{(e_i - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,i})^2}{Q_{ii}} \right\}, \tag{49}$$

where the index set $\mathcal{A}$ is

$$\mathcal{A} = \left\{ i : \alpha_i > 0, \text{ if } \frac{e_i - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,i}}{Q_{ii}} < 0 \text{ or } \alpha_i < C, \text{ if } \frac{e_i - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,i}}{Q_{ii}} > 0 \right\}. \tag{50}$$

6.2 The clipDCD algorithm for RKWMTSVM

We embed the clipDCD algorithm into the dual QPPs in our RKWMTSVM during each iteration of CCCP procedure. The two convex QPPs are shown in (36) and (37). For simplicity, we only explain the QPP (36), where we use matrix $\mathbf{N}_+(\mathbf{X}_1^T\mathbf{D}_1\mathbf{X}_1)^{-1}\mathbf{N}_+$ in place of matrix $\boldsymbol{Q}$ in (47) and use vector $\boldsymbol{\gamma}_+ - \boldsymbol{\tau}_+$ in place of $\boldsymbol{\alpha}$ in (47). Therefore, we need to solve the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{Q}\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ &s.t. \quad -\boldsymbol{\tau}_+ \le \boldsymbol{\alpha} \le \mathbf{S}_+ - \boldsymbol{\tau}_+. \end{aligned} \tag{51}$$

It is noteworthy the matrix $\mathbf{N}_+ = [\mathbf{X}_2^T\mathbf{F}_2;\ \mathbf{X}_3^T\mathbf{H}]$ is sparse, where $\mathbf{F}_2 = diag(f_1, f_2, ..., f_{l_2})$ and $\mathbf{H} = diag(h_1, h_2, ..., h_{l_3})$, because the value of $f_j, j \in I_2$ or $h_k, k \in I_3$ is either 1 or 0. That means the component of $\boldsymbol{\alpha}$ corresponding $f_j = 0$ and $h_k = 0$ will not contribute to the optimization of objective function. Therefore, they can be ignored during the progress of solving the optimal $\boldsymbol{\alpha}$, which will further advance the computational efficiency.

Remarkably, there is the simple update $\alpha_L^{new} = \alpha_L + \lambda$ in clipDCD. Then,

$$f(\lambda) = f(0) + \frac{1}{2}\lambda^2 \boldsymbol{Q}_{LL} - \lambda(e_L - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,L}) \tag{52}$$

Setting the derivation of λ:

$$\frac{df(\lambda)}{d\lambda} = 0 \quad \Rightarrow \lambda = \frac{e_L - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,L}}{\boldsymbol{Q}_{LL}}. \tag{53}$$

Similarly, the L index is chosen as Eq.(49). Moreover, the λ value in QPP (36) must be adequately clipped so that $-\boldsymbol{\tau}_+^{(i)} \le \alpha_L^{new} \le \mathbf{S}_+^{(i)} - \boldsymbol{\tau}_+^{(i)}$. That means the step λ should satisfy the inequality constraints. Accordingly, we adjust the index set $\mathcal{A}$ as follows.

$$\begin{aligned} \mathcal{A} = \{ i : &\alpha_i > -\boldsymbol{\tau}_+^{(i)}, \text{ if } \frac{e_i - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,i}}{Q_{ii}} < 0 \\ &\text{or } \alpha_i < \mathbf{S}_+^{(i)} - \boldsymbol{\tau}_+^{(i)}, \text{ if } \frac{e_i - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,i}}{Q_{ii}} > 0 \}. \end{aligned} \tag{54}$$

Table 5 Performance comparison of four algorithms on seven datasets

Dataset	Metrics	KNN	KWMTSVM	RKWMTSVM^{qp}	RKWMTSVM^{cl}
Soybean	Acc(%)±std	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**
	Time (s)	0.0034	0.810	0.0836	0.0435
Wine	Acc(%)±std	96.22±3.63	96.76±4.44	**97.84±3.52**	**97.84±2.96**
	Time (s)	0.0028	0.0908	0.1083	0.0717
Ecoli	Acc(%)±std	86.87±7.11	**88.36±4.99**	**88.36±3.06**	87.69±3.93
	Time (s)	0.0074	0.9149	1.2654	0.9247
Glass	Acc(%)±std	92.50±8.54	93.33±6.97	94.17±6.32	**95.14±2.41**
	Time (s)	0.0070	1.0065	1.1727	0.6725
Balance	Acc(%)±std	74.13±6.93	93.02±4.40	**93.17±4.07**	91.43±7.09
	Time (s)	0.0086	2.0982	3.0316	2.1261
Waveform	Acc(%)±std	80.44±2.13	85.11±1.77	**86.22±1.69**	85.56±1.76
	Time (s)	0.0123	5.1298	5.8813	4.6658
Pageblock	Acc(%)±std	94.58±1.15	94.00 ±1.05	**95.07±1.73**	93.59±1.63
	Time (s)	0.0810	745.1607	1070.8629	904.384

The bold values indicate the best predictive accuracies

Table 6 Performance comparison of four algorithms on seven datasets with 5% noises

Dataset	Metrics	KNN	KWMTSVM	RKWMTSVM^{qp}	RKWMTSVM^{cl}
Soybean	Acc(%)±std	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**
	Time (s)	0.0041	0.1099	0.1197	0.0437
Wine	Acc(%)±std	95.14±4.83	94.59±4.27	95.68±5.60	**96.22±4.52**
	Time (s)	0.004	0.1207	0.1288	0.1125
Ecoli	Acc(%)±std	85.67±4.90	85.37±5.52	**88.06±4.09**	86.27±5.52
	Time (s)	0.0062	1.5984	2.0571	1.8183
Glass	Acc(%)±std	78.75±9.48	78.33±10.98	**85.42±6.07**	79.58±6.49
	Time (s)	0.0072	1.3218	1.3499	0.2344
Balance	Acc(%)±std	**79.21±4.97**	76.67±6.11	76.98±7.82	75.40±1.37
	Time (s)	0.0074	2.3787	5.8000	2.8943
Waveform	Acc(%)±std	80.56±2.64	82.44±2.50	**84.56±2.90**	83.33±2.08
	Time (s)	0.0120	4.9974	4.4942	1.3547
Pageblock	Acc(%)±std	93.43±1.57	93.19±0.54	**93.65±1.29**	93.34±0.96
	Time (s)	0.0611	1867.6467	2228.8306	1813.1565

The bold values indicate the best predictive accuracies

The convergence of clipDCD algorithm and corresponding theoretical proofs can be found in Peng et al (2014).

In a word, the framework of clipDCD algorithm for solving QPP(36) is summarized as follows.

Algorithm 2 ClipDCD for solving QPP(36).

```
1. Initialize $\boldsymbol{\alpha} = \mathbf{0}$.
2. While $\boldsymbol{\alpha}$ is not optimal;
1) Choose the $L$ index by (49) and (54), and compute $\lambda$ by Eq().
2) Update $\alpha_L$ as $\alpha_L^{new} = [\alpha_L + \lambda]_\sharp$, where $[u]_\sharp = max(-\boldsymbol{\tau}_+^{(i)}, min(u, \mathbf{S}_+^{(i)} - \boldsymbol{\tau}_+^{(i)}))$.
```

In order to verify the performance of our proposed RKWMTSVM by applying the clipDCD (RKWMTSVM^{cl}), we conduct experiment on seven benchmark datasets. Besides, we add noises into these datasets. The corresponding results are recorded in Tables 5 and 6, respectively. The results indicate that the clipDCD algorithm can improve the solving speed of our RKWMTSVM to a certain extent, while the testing accuracy lose a little. The results also show that the proposed RKWMTSVM by quadratic programming method (RKWMTSVM^{qp}) obtain the best testing accuracy on most datasets. Although the RKWMTSVM^{cl} performs almost the same as KWMTSVM in Table 5, it performs better than KWMTSVM after the noises are added in Table 6. In addition, we also compare our method with the native multi-class classifier K-Nearest Neighbor (KNN). The results show that our RKWMTSVM outperforms KNN from the perspective of testing accuracy, while it takes a relatively long time.

7 Conclusion and future work

In this paper, a new multi-classification algorithm termed as ramp loss K-nearest neighbor-weighted multi-class twin support vector machine (RKWMTSVM) is proposed by replacing the hinge loss with ramp loss function in KWMTSVM. This modification leads to a precise, sparse and robust algorithm with better performance. On one hand, similar to KWMTSVM, the proposed RKWMTSVM adopts the OVOVR structure to deal with multi-class classification problem, which is helpful to deal with class-imbalance problem by considering all training points. Besides, the local information of intra-class can be exploited by constructing the weight matrix in the objective function which can help to improve the generalization ability. Meanwhile, the weight matrix of inter-class can help to remove the redundant constraints which can reduce the computational speed. On the other hand, it adopts the ramp loss function which can avoid the disturbance of outliers, making it be a sparser and robust algorithm, especially compared with KWMTSVM. In addition, the proposed RKWMTSVM is a non-differentiable non-convex optimization problem, we adopt the CCCP to cope with this model and embed the clipDCD method to speed up the solution process, thus making our model more adaptable for large-scale problem. In the experiments, we compare it with six state-of-the-art algorithms to demonstrate the validity. The experimental results show that the proposed RKWMTSVM is a more sparse algorithm and is robust to outliers.

As the proposed RKWMTSVM is less sensitive to outliers and it is an efficient multi-classification algorithm, it is worthwhile to apply it on the domains such as intrusion detection problem, spam detection, stellar spectra classification, fault class detection, novelty detection, prediction of diagnostic phenotypes, customer churning analysis, anomaly detection problem and so on. Due to the proposed RKWMTSVM contains many parameters, it takes us a long searching time to find the optimal parameters combination by grid search. During our experiments, we find that the performance of RKWMTSVM is worse on some parameter intervals which is time-consuming and wasteful to search the optimal solution on these parameter intervals. Besides, the optimal solution is not always fall in the grid points of grid search. Therefore, how to design an effective algorithm for searching the optimal parameter combination for our RKWMTSVM is worthy of further research.

Acknowledgements The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

Funding This work was supported in part by the Fundamental Research Funds for the Central Universities (No. 2021ZY92, BLX201928), National Natural Science Foundation of China (No.11671010) and Beijing Natural Science Foundation (No. 4172035).

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest Author Huiru Wang declares that she has no conflict of interest. Author Yitian Xu declares that he has no conflict of interest. Author Zhijian zhou declares that she has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

An Y, Xue H (2022) Indefinite twin support vector machine with dc functions programming. Pattern Recognit 121:108195

Angulo C, Parra X, Català A (2003) K-svcr. A support vector machine for multi-class classification. Neurocomputing 55(1–2):57–77

Balasundaram S, Gupta D, Kapil (2014) 1-norm extreme learning machine for regression and multiclass classification using newton method. Neurocomputing 128:4–14

Bamakan SMH, Wang H, Shi Y (2017) Ramp loss k-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem. Knowl Based Syst 126:113–126

Borah P, Gupta D (2020) Functional iterative approaches for solving support vector classification problems based on generalized huber loss. Neural Comput Appl 32(13):9245–9265

Borah P, Gupta D (2021) Robust twin bounded support vector machines for outliers and imbalanced data. Appl Intell 51:5314–5343

Deepak G, Bharat R, Parashjyoti B (2018) A fuzzy twin support vector machine based on information entropy for class imbalance learning. Neural Comput Appl 31:1–12

Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7(1):1–30

Friedrichs F, Igel C (2005) Evolutionary tuning of multiple svm parameters. Neurocomputing 64(1):107–117

Gupta D (2017) Training primal k-nearest neighbor based weighted twin support vector regression via unconstrained convex minimization. Appl Intell 47(3):1–30

Gupta D, Richhariya B (2018) Entropy based fuzzy least squares support vector machine for class imbalance learning. Appl Intell 48:4212–4231

Gutierrez PA, Perez-Ortiz M, Sanchez-Monedero J, Fernandez-Navarro F, Hervas-Martinez C (2016) Ordinal regression methods: survey and experimental study. IEEE T Knowl Data En 28(1):127–146

Hamdia KM, Ghasemi H, Zhuang X, Alajlan N, Rabczuk T (2018) Sensitivity and uncertainty analysis for flexoelectric nanostructures. Comput Method Appl M 337:95–109

Hazarika BB, Gupta D (2021) Density-weighted support vector machines for binary class imbalance learning. Neural Comput Appl 33:4243–4261

Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6(2):65–70

Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Netw 13(2):415–425

Huang X, Shi L, Suykens JAK (2014a) Ramp loss linear programming support vector machine. J Mach Learn Res 15(1):2185–2211

Huang X, Shi L, Suykens JAK (2014b) Support vector machine classifier with pinball loss. IEEE Trans Pattern Anal Mach Intell 36(5):984–997

Jayadeva KR, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910

Jha S, Mehta AK (2020) A hybrid approach using the fuzzy logic system and the modified genetic algorithm for prediction of skin cancer. Neural Process Lett

Kumar D, Thakur M (2018) All-in-one multicategory least squares nonparallel hyperplanes support vector machine. Pattern Recogn Lett 105:165–174

Kumar MA, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36(4):7535–7543

Li S, Fang H, Liu X (2018) Parameter optimization of support vector regression based on sine cosine algorithm. Expert Syst Appl 91:63–77

Lipp T, Boyd S (2016) Variations and extension of the convex-concave procedure. Optim Eng 17(2):263–287

Liu D, Shi Y, Tian Y (2015) Ramp loss nonparallel support vector machine for pattern classification. Knowl Based Syst 85:224–233

Liu D, Shi Y, Tian Y, Huang X (2016) Ramp loss least squares support vector machine. J Comput Sci 14:61–68

Long T, Yj T, Wj L, Pm P (2020) Structural improved regular simplex support vector machine for multiclass classification. Appl Soft Comput 91(106):235

Lu S, Wang H, Zhou Z (2019) All-in-one multicategory ramp loss maximum margin of twin spheres support vector machine. Appl Intell 49:2301–2314

Mir A, Nasiri JA (2018) Knn-based least squares twin support vector machine for pattern classification. Appl Intell 48(12):4551–4564

Nasiri JA, Moghadam CN, Jalili S (2015) Least squares twin multi-class classification support vector machine. Pattern Recognit 48(3):984–992

Ortigosa HJ, Inza I, Lozano JA (2017) Measuring the class-imbalance extent of multi-class problems. Pattern Recogn Lett 98:32–38

Pan X, Luo Y, Xu Y (2015) K-nearest neighbor based structural twin support vector machine. Knowl Based Syst 88:34–44

Peng X (2011) Building sparse twin support vector machine classifiers in primal space. Inf Sci 181(18):3967–3980

Peng X (2011) Tpmsvm: a novel twin parametric-margin support vector machine for pattern recognition. Pattern Recognit 44(10):2678–2692

Peng X, Chen D (2018) Ptsvrs: regression models via projection twin support vector machine. Inf Sci 435(1):1–14

Peng X, Chen D, Kong L (2014) A clipping dual coordinate descent algorithm for solving support vector machines. Knowl Based Syst 71:266–278

Peng X, Xu D, Kong L, Chen D (2016) L1-norm loss based twin support vector machine for data recognition. Inf Sci 340–341:86–103

Prasad SC, Balasundaram S (2021) On lagrangian l2-norm pinball twin bounded support vector machine via unconstrained convex minimization. Inf Sci 571:279–302

Qi Z, Tian Y, Yong S (2013) Robust twin support vector machine for pattern classification. Pattern Recogn 46(1):305–316

Qi Z, Tian Y, Shi Y (2014) A nonparallel support vector machine for a classification problem with universum learning. J Comput Appl Math 263:288–298

Rastogi R, Pal A, Chandra S (2018) Generalized pinball loss svms. Neurocomputing 322:151–165

Rezvani S, Wang X (2021) Class imbalance learning using fuzzy art and intuitionistic fuzzy twin support vector machines. Inf Sci 278:659–682

Richhariya B, Tanveer M (2020) A reduced universum twin support vector machine for class imbalance learning. Pattern Recognit 102(107):150

Salvador G, Alberto F, Luengo J, Francisco H (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Inf Sci 180(10):2044–2064

Salzberg SL (1997) On comparing classifiers: pitfalls to avoid and a recommended approach. Data Min Knowl Disc 1(3):317–328

Schölkopf B, Platt J, Hofmann, T (2007) An efficient method for gradient-based adaptation of hyperparameters in SVM Models. Adv Neural Inf Process Syst 19: Proceedings of the 2006 Conference, MIT Press, 673–680

Sharma S, Rastogi R, Chandra S (2021) Large-scale twin parametric support vector machine using pinball loss function. IEEE T Syst Man CY-S 51(2):987–1003

Shi Y (2012) Twin support vector machine with universum data. Neural Netw 36:112–119

Simes RJ (1986) An improved bonferroni procedure for multiple tests of significance. Biometrika 73(3):751–754

Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–330

Tang L, Tian Y, Li W, Pardalos PM (2021) Valley-loss regular simplex support vector machine for robust multiclass classification. Knowl-Based Syst 216(3):106–801

Tanveer M (2014) Robust and sparse linear programming twin support vector machines. Cogn Comput 7(1):137–149

Tanveer M, Aruna T, Rahul C, Jalan S (2019) Sparse pinball twin support vector machines. Appl Soft Comput 78:164–175

Tanveer M, Gautam C, Suganthan PN (2019) Comprehensive evaluation of twin svm based classifiers on uci datasets. Appl Soft Comput 83:105617

Tanveer M, Sharma A, Suganthan PN (2019) General twin support vector machine with pinball loss function. Inf Sci 494:311–327

Tanveer M, Sharma A, Suganthan PN (2021) Least squares knn-based weighted multiclass twin svm. Neurocomputing 459:454–464

Tanveer M, Sharma S, Rastogi R (2021) Sparse support vector machine with pinball loss. T Emerg Telecommun T 32(2):e3820

Tharwat A, Hassanien AE, Elnaghi BE (2017) A ba-based algorithm for parameter optimization of support vector machine. Pattern Recogn Lett 93:13–22

Tian Y, Ju X, Qi Z (2014) Efficient sparse nonparallel support vector machines for classification. Neural Comput Appl 24(5):1089–1099

Vapnik VN (1995) The nature of statistical learning theory. Springer, Berlin

Wang H, Zhou Z (2017) An improved rough margin-based ν-twin bounded support vector machine. Knowl Based Syst 128:125–138

Wang H, Lu S, Zhou Z (2020) Ramp loss for twin multi-class support vector classification. Int J Syst Sci 51(8):1448–1463

Wang Z, Shao YH, Wu TR (2013) A ga-based model selection for smooth twin parametric-margin support vector machine. Pattern Recognit 46(8):2267–2277

Xiao Y, Wang H, Xu W (2017) Ramp loss based robust one-class svm. Pattern Recogn Lett 85:15–20

Xu Y (2016) K-nearest neighbor-based weighted multi-class twin support vector machine. Neurocomputing 205:430–438

Xu Y, Wang L, Zhong P (2012) A rough margin-based ν-twin support vector machine. Neural Comput Appl 21:1307–1317

Xu Y, Guo R, Wang L (2013) A twin multi-class classification support vector machine. Cogn Comput 5(4):580–588

Xu Y, Yu J, Zhang Y (2014) Knn-based weighted rough v-twin support vector machine. Knowl Based Syst 71:303–313

Ye Q, Zhao C, Gao S, Zheng H (2012) Weighted twin support vector machines with local information and its application. Neural Netw 35:31–39

Yuille A, Rangarajan A (2003) The concave-convex procedure. Neural Comput 14(4):915–936

Zhou L, Wang Q, Fujita H (2017) One versus one multi-class classification fusion using optimizing decision directed acyclic graph for predicting listing status of companies. Inform Fusion 36:80-89

Zhu F, Yang J, Gao C, Xu S, Ye N, Yin T (2016) A weighted one-class support vector machine. Neurocomputing 189(12):1–10