CrossMark

KNN-based least squares twin support vector machine for pattern classification

A. Mir[1] · Jalal A. Nasiri[2]

Abstract

The least squares twin support vector machine (LSTSVM) generates two non-parallel hyperplanes by directly solving a pair of linear equations as opposed to solving two quadratic programming problems (QPPs) in the conventional twin support vector machine (TSVM), which makes learning speed of LSTSVM faster than that of the TSVM. However, LSTSVM fails to discover underlying similarity information within samples which may be important for classification performance. To address the above problem, we apply the similarity information of samples into LSTSVM to build a novel non-parallel plane classifier, called K-nearest neighbor based least squares twin support vector machine (KNN-LSTSVM). The proposed method not only retains the superior advantage of LSTSVM which is simple and fast algorithm but also incorporates the inter-class and intra-class graphs into the model to improve classification accuracy and generalization ability. The experimental results on several synthetic as well as benchmark datasets demonstrate the efficiency of our proposed method. Finally, we further went on to investigate the effectiveness of our classifier for human action recognition application.

Keywords Classification · Twin support vector machine · Least squares · K-nearest neighbor · Similarity information

1 Introduction

Support vector machine [6] is a powerful method for pattern classification and regression. It has been successfully applied in a wide variety of real-world problems such as face recognition [28], voice classification [33], text categorization [20], bioinformatics [25] and civil engineering [27]. SVM is on the basis of Vapnik-Chervonekis (VC) dimension and structural risk minimization. Its basic idea is to find the optimal separating hyperplane between positive and negative samples which involves solving of a convex quadratic programming problem (QPP).

Researchers have made many improvements on the basis of SVM. For instance, Fung and Mangasrian [22] proposed proximal support vector machine (PSVM) for binary classification. This method generates two parallel hyperplanes for each class of data points. That is, each plane must be as close as possible to its class and as far as possible from other class. Following PSVM, Mangasrian and Wild [23] proposed generalized eigenvalue proximal SVM (GEPSVM) which does binary classification by obtaining two non-parallel hyperplanes. In this method, samples of each class are proximal to one of two non-parallel planes. Two non-parallel planes are obtained by solving the eigenvector corresponding to the smallest eigenvalue of a generalized eigenvalue problem.

Jayadeva et al. [12] proposed twin support vector machine (TSVM) for binary classification, inspired by GEPSVM. TSVM seeks two non-parallel hyperplanes such that each hyperplane is as close as possible to samples of its own class and as far as possible from samples of other class. The main idea is to solve two smaller QPPs rather than a single large QPP which makes training speed of TSVM four times faster than that of a standard SVM. The experimental results in [12] show the superiority of TSVM over SVM and GEPSVM on UCI datasets.

In the last decade, TSVM has been enhanced rapidly [8, 9, 13, 38]. Some improvements have been made to TSVM by researchers to obtain higher classification accuracy with lower computational time such as Least Squares

✉ Jalal A. Nasiri
   j.nasiri@irandoc.ac.ir

   A. Mir
   mir-am@hotmail.com

1   Faculty of Computer Engineering, Islamic Azad University, North Tehran Branch, Tehran, Iran

2   Iranian Research Institute for Information Science and Technology (IranDoc), Tehran, Iran

Twin Support Vector Machine (LSTSVM), Twin Bounded Support Vector Machine (TBSVM), Weighted TwinSVM with local information (WLTSVM), Energy-based Least Squares TwinSVM (ELS-TSVM), Robust Energy-based Twin Support Vector Machines (RELS-TSVM), Angle-based Twin Support Vector Machine (ATSVM) and An Improved v-Twin Bounded Support Vector Machine (Iv-TBSVM) [15–17, 26, 31, 35, 37, 41, 42].

The major disadvantage of TSVM is that it fails to exploit the similarity information between any pair of samples that may be important for classification. Yi et al. [42] proposed WLTSVM which considers the similarity information between pairs of samples by finding the k-nearest neighbors for all the samples. This approach has three additional advantages over TSVM: (1) close or better classification accuracy compared to TSVM. (2) Different from TSVM, it considers marginal points of each class instead of all the data points. (3) It has only one penalty parameter.

Similar to TSVM, LSTSVM [17] fails to exploit the similarity information between any pair of samples. However, LSTSVM is simple, extremely fast and solves two systems of linear equations as opposed to solving two QPPs in TSVM. In this paper, we present an improvement on LSTSVM and embed the similarity information between pairs of samples into the optimization problems of LSTSVM based on k-nearest neighbor graph. The proposed method possesses the following advantages:

1. Similar to WLTSVM, the proposed method (KNN-LSTSVM) makes full use of similarity information between pairs of samples. That is, it uses k-nearest neighbor graph to characterize the intra-class compactness and inter-class separability, respectively. Based on this, KNN-LSTSVM achieves higher classification accuracy and better generalization ability.
2. Unlike WLTSVM, KNN-LSTSVM solves two systems of linear equations which leads to simple algorithm and less computational time. As a result, the proposed method does not need any external optimizer.
3. The LS-TSVM is more sensitive to the outliers. However, KNN-LSTSVM gives much less weight to the outliers. As a result, the final hyperplane is less sensitive to the outliers and noisy samples and is potentially robust.

Numerical experiments on several synthetic and benchmark datasets show that our KNN-LSTSVM obtains better classification ability in comparison with TSVM, WLTSVM and LSTSVM. We also explored the application of linear KNN-LSTSVM to human action recognition problem. It is the task of assigning a given video to one of action categories. Moreover, human action recognition has several challenges such as intra-class and inter-class variations, environment settings and temporal variations [30].

This paper is organized as follows. Section 2 briefly introduces SVM and TSVM. Section 3 gives the details of KNN-LSTSVM, including linear and non-linear cases. Section 4 discusses the experimental results on various datasets to investigate the effectiveness and validity of our proposed method. Finally, concluding remarks are given in Section 5.

2 Background

In this section, we briefly explain the basics of SVM and TSVM.

2.1 Support vector machine

Consider the binary classification problem with the training set $T = \{(x_1, y_1), ..., (x_n, y_n)\}$, where are $x_i \in \mathbb{R}^n$ feature vectors in the n-dimensional real space and $y_i \in \{-1, 1\}$ are the corresponding labels. The idea is to find a separating hyperplane $w^T x + b = 0$ where $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Figure 1 shows geometric interpretation of standard SVM for a toy dataset.

Assume that all the samples are strictly linearly separable. Then, the optimal separating hyperplane is obtained by solving the following optimization problem.

$$\min_w \frac{1}{2}\|w\|^2$$
$$\text{s.t. } y_i(w^T x_i + b) \geq 1, \forall i \tag{1}$$

The problem defined by (1) is called 'hard margin' and the inequality constraint should be satisfied for all the samples. In real world scenarios, two classes are not often linearly separable. Therefore, an error occurs in satisfying the inequality for some samples. The slack variable added
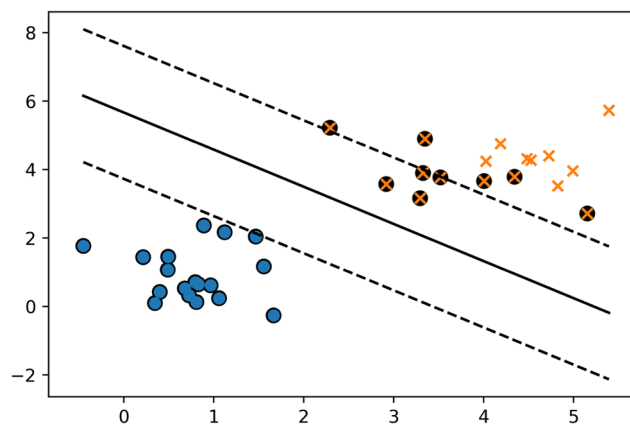

Fig. 1 Geometric interpretation of standard SVM

to the problem (2) to classify samples with some error. The problem (2) is modified as

$$\min_w \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{s.t. } y_i(w^Tx_i + b) + \xi_i \geq 1, \forall i \tag{2}$$

Where ξ_i is a slack variable associated with x_i sample and C is a penalty parameter. In this case, the classifier is known as soft margin. The Wolfe dual of (2) is given by

$$\min_\alpha \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n}\alpha_i$$
$$\text{s.t. } \sum_{i=1}^{n}\alpha_i y_i = 0, \ 0 \leq \alpha_i \leq C, \ \forall i \tag{3}$$

Where $\alpha \in \mathbb{R}$ are Lagrange multipliers. The parameters of optimal separating hyperplane are given by

$$w = \sum_{i=1}^{n}\alpha_i^* y_i x_i \tag{4}$$

Where α^* is the solution of the dual problem (3). A new sample is classified as +1 and -1 according to the decision function $D(x) = sign(w^Tx + b)$. SVM can be extended in a simple manner to handle nonlinear kernels [36].

2.2 Twin support vector machine

Consider a binary classification problem of m_1 positive samples and m_2 negative samples ($m_1 + m_2 = m$). Suppose that samples in class +1 are denoted by a matrix $A \in \mathbb{R}^{m_1 \times n}$, where each row represents a sample. Similarly, the matrix $B \in \mathbb{R}^{m_2 \times n}$ represents the samples of class -1. Unlike SVM, TSVM does classification using two non-parallel hyperplanes:

$$x^T w^{(1)} + b^{(1)} = 0, x^T w^{(2)} + b^{(2)} = 0 \tag{5}$$

where $w^{(1)}, w^{(2)} \in \mathbb{R}^n$ and $b^{(1)}, b^{(2)} \in \mathbb{R}$. Each hyperplane is close to samples of one class and far from the samples
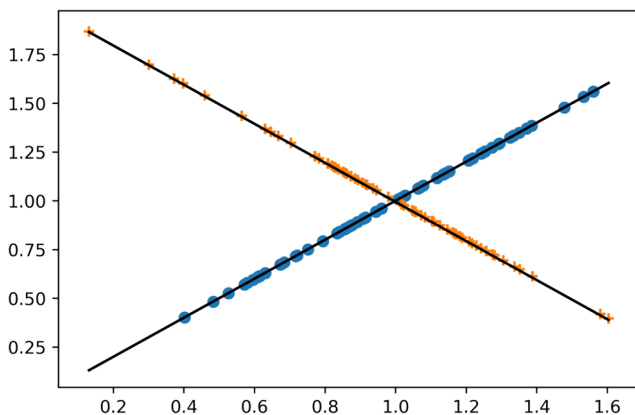

Fig. 2 Geometric interpretation of standard TSVM

of other class. Figure 2 shows geometric interpretation of standard TSVM for crossplane dataset.

TSVM solves the following two QPPs for obtaining two non-parallel hyperplanes:

$$\min_{w_{(1)},b_{(1)}} \frac{1}{2}\|Aw^{(1)} + e_1 b^{(1)}\|^2 + C_1 e_2^T y$$
$$\text{s.t. } -(Bw^{(1)} + e_2 b^{(1)}) + y \geq e_2, \ y \geq 0 \tag{6}$$

$$\min_{w_{(2)},b_{(2)}} \frac{1}{2}\|Bw^{(2)} + e_2 b^{(2)}\|^2 + C_2 e_1^T y$$
$$\text{s.t. } (Aw^{(2)} + e_1 b^{(2)}) + y \geq e_1, \ y \geq 0 \tag{7}$$

where $C_1, C_2 \geq 0$ are penalty parameters, e_1, e_2 are vectors of ones of appropriate dimensions. It can be noted that samples of one class appear in the constraints of each QPP. As a result, TSVM is almost four times faster than a standard SVM. By introducing Lagrange multipliers, the Wolf dual of QPPs (6) and (7) are represented as follows:

$$\min_\alpha \frac{1}{2}\alpha^T G(H^T H)^{-1}G^T\alpha - e_2^T\alpha$$
$$\text{s.t. } 0e_2 \leq \alpha \leq C_1 e_2 \tag{8}$$

$$\min_\beta \frac{1}{2}\beta^T P(Q^T Q)^{-1}P^T\beta - e_1^T\beta$$
$$\text{s.t. } 0e_1 \leq \beta \leq C_2 e_1 \tag{9}$$

where $H = [A \ e]$, $G = [B \ e]$, $P = [A \ e]$ and $Q = [B \ e]$, $\alpha \in \mathbb{R}^{m_2}$ and $\beta \in \mathbb{R}^{m_1}$ are Lagrangian multipliers. The two dual QPPs (8) and (9) have the advantage of bounded constraints and reduced number of parameters, implying that QPP (8) has only m_2 parameters and QPP (9) has only m_1 parameters.

The two non-parallel hyperplanes (5) can be obtained from the solution of QPP (8) and (9) by

$$\begin{bmatrix} w^{(1)} \\ b^{(1)} \end{bmatrix} = -(H^T H)^{-1}G^T\alpha \tag{10}$$

$$\begin{bmatrix} w^{(2)} \\ b^{(2)} \end{bmatrix} = (Q^T Q)^{-1}P^T\beta \tag{11}$$

The matrices $H^T H$ and $Q^T Q$ are of size $(n+1) \times (n+1)$ where $n \ll m$. Once the two hyperplanes (5) are obtained, a new sample is assigned to class i ($i = +1, -1$) depending on which of the two hyperplanes in (5) it is closer to

$$Class \ i = \underset{j=1,2}{arg \min} \left| x^T w^{(j)} + b^{(j)} \right| \tag{12}$$

where $|.|$ is perpendicular distance. The case of non-linear kernels is handled similar to linear kernels [12]. According to [42] TSVM has the following two limitations:

- It fails to exploit the similarity information between pairs of samples. Previous studies have shown that most of the samples are highly correlated [42]. Therefore, underlying similarity is crucial for data classification [4].
- In TSVM, hyperplane of each class should be far from all the samples of other class. Consequently,

TSVM may misclassify margin points. By utilizing a few margin points from other class instead of all the samples, TSVM may achieve better result [42].

WLTSVM addressed these limitations by making full use of similarity information in terms of the data affinity [42].

3 KNN-based least squares twin support vector machine

In this section, we present our proposed method(KNN-LSTSVM) and explain its details. Section 3.1 discusses construction of weight matrices. Details of linear KNN-LSTSVM are given in Section 3.2. Proposed method extended to nonlinear kernels in Section 3.3.

3.1 Construction of weight matrices

The central idea of WLTSVM and proposed method is to give larger weights to the samples with high density and extract possible margin points from the samples of other class. In other words, the hyperplane yielded by KNN-LSTSVM fits the samples with high density and are far from the margin points of other class. Figure 3 shows the geometrical interpretation of LSTSVM and KNN-LSTSVM for both linear and Gaussian kernel. As shown in Fig. 3, the hyperplane of KNN-LSTSVM is less sensitive to outliers and noisy samples. On the other hand, the hyperplane of LSTSVM is close to the outliers of circle class, because the hyperplane is as close as possible to all the samples of its own class. In order to demonstrate the main idea of KNN-LSTSVM clearly, the perpendicular distance of one margin point is calculated for both classifiers in Fig. 3.
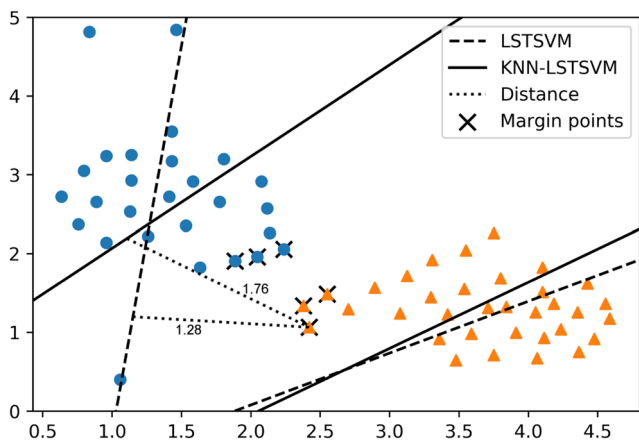

Fig. 3 The comparison of LSTSVM with KNN-LSTSVM

Similar to WLTSVM, a k-nearest neighbor graph is constructed as follows:

$$W_{ij} = \begin{cases} 1, & \text{if } x_i \in N\left(x_j\right) \text{ or } x_j \in N\left(x_i\right) \\ 0, & \text{otherwise.} \end{cases} \quad (13)$$

where the set $N(x_j)$ contains k-nearest neighbors of x_j and is given by: [4]

$$N(x_j) = \{x_j^1, x_j^2, \ldots, x_j^k\} \quad (14)$$

However, the graph W cannot reveal the discriminative structure in data [42]. Instead, a within-class graph W_w and between-class graph W_b are constructed to model the intra-class compactness and inter-class separability, respectively. The weight matrices W_w and W_b of class +1 are respectively defined as:

$$W_{w,ij} = \begin{cases} 1, & \text{if } x_i \in N_w(x_j) \text{ or } x_j \in N_w(x_i) \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

$$W_{b,ij} = \begin{cases} 1, & \text{if } x_i \in N_b(x_j) \text{ or } x_j \in N_b(x_i) \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

where $N_w(x_j)$ stands for k-nearest neighbors of x_j from same class and $N_b(x_j)$ contains k-nearest neighbors of x_j from different class. The two sets $N_w(x_j)$ and $N_b(x_j)$ are respectively given by:

$$N_w(x_i) = \{x_i^j \mid l(x_i^j) = l(x_i), 1 \leq j \leq k\} \quad (17)$$

$$N_b(x_i) = \{x_i^j \mid l(x_i^j) \neq l(x_i), 1 \leq j \leq k\} \quad (18)$$

where $l(x_i)$ denotes the class label of x_i. Clearly, $N_w(x_i) \cap N_b(x_i) = \varnothing$ and $N_w(x_i) \cup N_b(x_i) = N(x_i)$. The distance between pairs of samples is measured by using the standard Euclidean metric. In order to find the margin points of class -1, the weight matrix $W_{b,ij}$ is redefined as follows:

$$f_j = \begin{cases} 1, & \text{if } \exists i, W_{b,ij} \neq 0 \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

The weight of each sample in class +1 is computed by

$$d_j = \sum_{i=1}^{m_1} W_{w,\,ij}, \quad j = 1, 2, \ldots, m_1 \quad (20)$$

where d_j denotes the weight of x_j. The value of d_j shows how much dense x_j is. In other words, the number of neighbors with same label determines the weight of a sample.

3.2 Linear KNN-LSTSVM

Similar to WLTSVM, proposed method seeks a pair of non-parallel hyperplanes, each of which fits samples with high density and is far from the margin points of other class. However, KNN-LSTSVM modifies the

primal problems of WLTSVM by replacing the inequality constraints with equality constraints and using the square of 2-norm slack variables. The solution of two modified primal problems requires solving two systems of linear equations as opposed to solving two QPPs in WLTSVM. Also note that modification of primal problems leads to extremely fast and simple method for obtaining two non-parallel hyperplanes. The modified primal problem of class 1 can be expressed as follows:

$$
\min_{w_{(1)}, b_{(1)}} \quad \frac{1}{2}(Aw^{(1)} + eb^{(1)})^T D(Aw^{(1)} + eb^{(1)})
$$
$$
+ \frac{C}{2} y^T y \qquad (21)
$$
$$
\text{s.t.} \quad -F(Bw^{(1)} + eb^{(1)}) + y = Fe
$$

where $D = diag(d_1, d_2, \ldots, d_{m_1})$ and $F = diag(f_1, f_2, \ldots, f_{m_2})$ are the weight matrix of class +1 and the margin points of class -1, respectively. Cleary, f_j is either 0 or 1 and d_i is greater than or equal to zero ($d_i \geq 0$).

The modified primal problem (21) allows to substitute the equality constraints into the objective function, thus Lagrangian of (21) becomes:

$$
\min_{w_{(1)}, b_{(1)}} \quad L = \frac{1}{2} \left\| D(Aw^{(1)} + eb^{(1)}) \right\|^2
$$
$$
+ \frac{C}{2} \left\| F(Bw^{(1)} + eb^{(1)}) + Fe \right\|^2 \qquad (22)
$$

By taking partial derivative of (22) with respect to $w^{(1)}$ and $b^{(1)}$, gives:

$$
\frac{\partial L}{\partial w^{(1)}} = A^T D(Aw^{(1)} + eb^{(1)})
$$
$$
+ CB^T F(Bw^{(1)} + eb^{(1)} + Fe) = 0e \qquad (23)
$$

$$
\frac{\partial L}{\partial b^{(1)}} = e^T D(Aw^{(1)} + eb^{(1)})
$$
$$
+ Ce^T F(Bw^{(1)} + eb^{(1)} + e) = 0 \qquad (24)
$$

Next, combining (23) and (24) and solving for $w^{(1)}$ and $b^{(1)}$, leads to a system of linear equations which is expressed as follows:

$$
\begin{bmatrix} B^T FB & B^T Fe \\ e^T FB & e^T Fe \end{bmatrix} \begin{bmatrix} w^{(1)} \\ b^{(1)} \end{bmatrix} + \frac{1}{C} \begin{bmatrix} A^T DA & A^T De \\ e^T DA & e^T De \end{bmatrix} \begin{bmatrix} w^{(1)} \\ b^{(1)} \end{bmatrix}
$$
$$
+ \begin{bmatrix} B^T Fe \\ e^T Fe \end{bmatrix} = 0e. \quad (25)
$$

$$
\begin{bmatrix} w^{(1)} \\ b^{(1)} \end{bmatrix} = \begin{bmatrix} B^T FB + \frac{1}{C} A^T DA & B^T Fe + \frac{1}{C} A^T De \\ e^T FB + \frac{1}{C} e^T DA & e^T Fe + \frac{1}{C} e^T De \end{bmatrix}^{-1}
$$
$$
\times \begin{bmatrix} -B^T Fe \\ -e^T Fe \end{bmatrix} \qquad (26)
$$

$$
\begin{bmatrix} w^{(1)} \\ b^{(1)} \end{bmatrix} = \left[\begin{bmatrix} B^T \\ e^T \end{bmatrix} F \begin{bmatrix} B & e \end{bmatrix} + \frac{1}{C} \begin{bmatrix} A^T \\ e^T \end{bmatrix} D \begin{bmatrix} A & e \end{bmatrix} \right]^{-1}
$$
$$
\times \left[\begin{bmatrix} -B^T \\ -e^T \end{bmatrix} Fe \right] \qquad (27)
$$

By defining $H = [A\ e]$ and $G = [B\ e]$, the solution of minimization problem (21) becomes:

$$
\begin{bmatrix} w^{(1)} \\ b^{(1)} \end{bmatrix} = -\left(G^T FG + \frac{1}{C} H^T DH \right)^{-1} G^T Fe \qquad (28)
$$

The modified primal problem of class -1 can be represented as follows:

$$
\min_{w_{(2)}, b_{(2)}} \quad \frac{1}{2}(Bw^{(2)} + eb^{(2)})^T Q(Bw^{(2)} + eb^{(2)}) + \frac{C}{2} y^T y
$$
$$
\text{s.t.} \quad P(Aw^{(2)} + eb^{(2)}) + y = Pe
$$
$$
(29)
$$

where $Q = diag(q_1, q_2, \ldots, q_{m_2})$ and $P = diag(p_1, p_2, \ldots, p_{m_1})$ are the weight matrix of class -1 and the margin points of class +1, respectively. Similar to F matrix, the p_j is either 0 or 1. In similar way, the solution of primal problem (29) can be obtained as follows:

$$
\begin{bmatrix} w^{(2)} \\ b^{(2)} \end{bmatrix} = \left(H^T PH + \frac{1}{C} G^T QG \right)^{-1} H^T Pe \qquad (30)
$$

The solutions of (28) and (30) involves two matrix inverses of size $(n+1) \times (n+1)$ where n is much smaller than the number samples of classes 1 and -1. Consequently, the learning speed of linear KNN-LSTSVM is extremely fast.

The decision function for assigning a class $i(i = +1, -1)$ to a new sample x_i is defined as follows:

$$
D(x_i) = \begin{cases} +1, & \text{if } |x^T w^{(1)} + b^{(1)}| < |x^T w^{(2)} + b^{(2)}| \\ -1, & \text{otherwise.} \end{cases}
$$
$$
(31)
$$

where $|.|$ denotes perpendicular distance of a sample from the hyperplane. In summary, algorithm 3.1 shows the required steps for constructing linear KNN-LSTSVM classifier.

Algorithm 3.1 Linear KNN-LSTSVM classifier

Input: Samples of class +1 represented by matrix $A \in \mathbb{R}^{m_1 \times n}$ and samples of class -1 represented by $B \in \mathbb{R}^{m_2 \times n}$.

(i) Select neighborhood size k. Then, define the weight matrices W_w and W_b for classes +1 and -1 using (15) and (16).

(ii) Construct diagonal matrices D, F, Q and P. Also define $H = [A\ e]$ and $G = [B\ e]$.

(iii) Select a penalty parameter C. This parameter is usually selected based on validation.

(iv) Obtain parameters of two non-parallel hyperplanes using (28) and (30).

(v) Calculate perpendicular distances $|x^T w^{(1)} + b^{(1)}|$ and $|x^T w^{(2)} + b^{(2)}|$ for a new sample $x \in \mathbb{R}^n$.

(vi) Assign the sample to class +1 or -1 using (31).

3.3 Non-linear KNN-LSTSVM

Following the same idea, the linear KNN-LSTSVM can be extended to non-linear version by considering the following kernel-generated surfaces instead of planes:

$$K(x^T, \ C^T)u^{(1)} + b^{(1)} = 0 \text{ and } K(x^T, \ C^T)u^{(2)} + b^{(2)} = 0 \tag{32}$$

where $C = \begin{bmatrix} A \\ B \end{bmatrix}$ and K is any arbitrary kernel function. Similar to linear version, the primal problems of linear KNN-LSTSVM can be modified in the same way with inequality constraints replaced by equality constraints as expressed in (33) and (34).

$$\min_{u_{(1)}, b_{(1)}} \quad \frac{1}{2}(K(A, \ C^T)u^{(1)} + eb^{(1)})^T D(K(A, \ C^T)u^{(1)}$$

$$+ eb^{(1)}) + \frac{C}{2} y^T y$$

$$\text{s.t.} \quad -F(K(B, \ C^T)u^{(1)} + eb^{(1)}) + y = Fe \tag{33}$$

$$\min_{u_{(2)}, b_{(2)}} \quad \frac{1}{2}(K(B, C^T)u^{(2)} + eb^{(2)})^T Q(K(B, C^T)u^{(2)}$$

$$+ eb^{(2)}) + \frac{C}{2} y^T y$$

$$\text{s.t.} \quad P(K(A, C^T)u^{(2)} + eb^{(2)}) + y = Pe \tag{34}$$

where $K(A, C^T)$ and $K(B, C^T)$ are kernel matrices of sizes $m_1 \times m$ and $m_2 \times m$ respectively ($m = m_1 + m_2$). By substituting the constraints into objective function, The QPPs (33) and (34) become:

$$\min_{u_{(1)}, b_{(1)}} \quad \frac{1}{2} \left\| D(K(A, \ C^T)u^{(1)} + eb^{(1)}) \right\|^2$$
$$+ \frac{C}{2} \left\| F(K(B, \ C^T)u^{(1)} + eb^{(1)} + Fe) \right\|^2 \tag{35}$$

$$\min_{u_{(2)}, b_{(2)}} \quad \frac{1}{2} \left\| Q(K(B, \ C^T)u^{(2)} + eb^{(2)}) \right\|^2$$
$$+ \frac{C}{2} \left\| P(K(A, \ C^T)u^{(2)} + eb^{(2)} + Pe) \right\|^2 \tag{36}$$

In a similar way, the solutions of QPPs (35) and (36) can be obtained as follows:

$$\begin{bmatrix} u^{(1)} \\ b^{(1)} \end{bmatrix} = -(S^T FS + \frac{1}{C} R^T DR)^{-1} S^T Fe \tag{37}$$

$$\begin{bmatrix} u^{(2)} \\ b^{(2)} \end{bmatrix} = (R^T PR + \frac{1}{C} S^T QS)^{-1} R^T Pe \tag{38}$$

where $R = \begin{bmatrix} K(A, C^T) & e \end{bmatrix}$ and $S = \begin{bmatrix} K(B, C^T) & e \end{bmatrix}$. A new sample is classified in the same way as it is done in linear case. The decision function for non-linear case is given by:

$$D(x) = \begin{cases} +1, & \text{if } |K(x, C^T)u^{(1)} + b^{(1)}| \\ & < |K(x, C^T)u^{(2)} + b^{(2)}| \\ -1, & \text{otherwise.} \end{cases} \tag{39}$$

It can be noted that the solution of non-liner KNN-LSTSVM requires inversion of matrix size $(m + 1) \times (m + 1)$ twice. However, using Sherman-Morrison-Woodbury (SMW) formula [10], the solution (37) and (38) can be solved using four inverses of smaller dimension than $(m + 1) \times (m + 1)$. The solution of (37) and (38) can be rewritten as:

$$\begin{bmatrix} u^{(1)} \\ b^{(1)} \end{bmatrix} = -(Y - YR^T D(CI + RYR^T D)^{-1} RY)S^T Fe \tag{40}$$

$$\begin{bmatrix} u^{(2)} \\ b^{(2)} \end{bmatrix} = (Z - ZS^T Q(CI + SZS^T Q)^{-1} SZ)R^T Pe \tag{41}$$

where $Y = (S^T FS)^{-1}$ and $Y = (R^T PR)^{-1}$. The matrix $(S^T FS)$ and $(R^T PR)$ might be positive semi-definite. Following [12], a regularization term εI, $\varepsilon > 0$ is introduced to Y and Z to avoid the possibility of the ill-conditioning of $(S^T FS)$ and $(R^T PR)$. This allows us to use SMW formula in finding Y and Z as

$$Y = \frac{1}{\varepsilon}(I - S^T F(\varepsilon I + SS^T F)^{-1} S) \tag{42}$$

$$Z = \frac{1}{\varepsilon}(I - R^T P(\varepsilon I + RR^T P)^{-1} R) \tag{43}$$

After using SMW formula, the solution of non-linear case requires two matrix inverses of size $(m_1 \times m_1)$ and two matrix inverses of size $(m_2 \times m_2)$. In summary, algorithm 3.2 shows the required steps for constructing non-linear KNN-LSTSVM classifier.

Algorithm 3.2 Non-linear KNN-LSTSVM classifier

Input: Samples of class +1 represented by matrix $A \in \mathbb{R}^{m_1 \times n}$ and samples of class -1 represented by $B \in \mathbb{R}^{m_2 \times n}$.

  (i)   Choose a kernel function K.
 (ii)   Select neighborhood size k. Then, define the weight matrices W_w and W_b for classes +1 and -1 using (15) and (16).
(iii)   Construct diagonal matrices D, F, Q and P. Also define $R = \begin{bmatrix} K(A, C^T) & e \end{bmatrix}$ and $S = \begin{bmatrix} K(B, C^T) & e \end{bmatrix}$.
 (iv)   Select a penalty parameter C. This parameter is usually selected based on validation.
  (v)   Obtain parameters of two non-parallel hyperplanes using (40) and (41).
 (vi)   Calculate perpendicular distances $|K(x, C^T)u^{(1)} + b^{(1)}|$ and $|K(x, C^T)u^{(2)} + b^{(2)}|$ for a new sample $x \in \mathbb{R}^n$.
(vii)   Assign the sample to class +1 or -1 using (39).

3.4 Discussion on KNN-LSTSVM

3.4.1 KNN-LSTSVM vs. LSTSVM

Compared with LSTSVM, KNN-LSTSVM incorporates KNN method into LSTSVM and embodies the similarity

Table 1 The characteristics of benchmark datasets

Datasets	#Samples	#Positive	#Negative	#Features
Austrailian	690	307	383	14
Bupa-Liver	345	145	200	6
Cleveland	303	139	164	13
Haber-Man	306	225	81	3
Heart-Statlog	270	120	150	13
Hepatits	155	32	123	19
Ionsphere	351	225	126	34
Monk3	554	288	266	6
Pima-Indian	768	268	500	8
Sonar	208	97	111	60
Titanic	891	342	549	7
Votes	435	267	168	16
Wdbc	569	212	357	30
Wpbc	198	47	151	33

information between pairs of samples into the objective function. Therefore proposed method is less sensitive to the outliers and noisy samples than LSTSVM. Both algorithms solve a pair of linear equations instead of two QPPs. As a result, they have fast learning speed.

3.4.2 KNN-LSTSVM vs. WLTSVM

Both WLTSVM and KNN-LSTSVM find two non-parallel hyperplanes by making full use of similarity information. They also give weight to the samples and have noise suppression capability. However, KNN-LSTSVM needs to solve two systems of linear equations as opposed to solving two QPPs in WLTSVM. It implies that the proposed method is faster than WLTSVM.

4 Numerical experiments

To demonstrate the performance of our proposed KNN-LSTSVM, we conducted experiments on 14 datasets from UCI machine learning repository.[1] They are Australian, Bupa-Liver, Cleveland, Haber-man, HeartStatlog, Hepatits, Ionsphere, Monk3, Pima-Indian, Sonar, Titanic, Votes, WDBC and WPBC. Table 1 shows the characteristics of these datasets.

Classification accuracy of each method is measured by standard 10-fold cross-validation. More specifically, each dataset is split into ten subsets randomly. One of those sets is reserved as a test set whereas the remaining data are considered for training. This process is repeated ten times until all of the ten subsets is tested [3].

4.1 Implementation details

All the methods were implemented in Python 3.6 programming language and ran on Windows 8 PC with Intel Core i7 6700K (4.0 GHZ) and 32.0 GB of RAM. In addition, NumPy package [40] was used for linear algebra operations and SciPy [14] package was employed for distance calculation and statistic functions. The solver is critical part of the code and is implemented in Cython[2] [2] which improves the training speed.

4.1.1 Optimizer

The proposed method solves two systems of linear equations for obtaining the hyperplanes. However, other algorithms such as standard SVM, standard TSVM and WLTSVM need an external optimizer for solving their QPPs problem. The clipDCD algorithm [29] was employed to solve the dual QPPs of these classifiers. This optimizer has simple formulation, fast learning speed and is easy-to-implement. It only solves one single-variable sub-problem according to the maximal possibility-decrease strategy [29]. The dual QPP of the standard SVM can be expressed as follows:

$$\min_{\alpha} \ \frac{1}{2}\alpha^T Q\alpha - e^T\alpha, \qquad \qquad (44)$$
$$\text{s.t.} \quad 0 \le \alpha \le C.$$

the clipDCD algorithm only updates one component of α at each iteration. More information on the convergence of this algorithm and theoretical proofs can be found in [29].

4.2 Parameters selection

The performance of TSVM and its extentions depend heavily on the choice of optimal parameters. Therefore, the grid search method is used to find the optimal parameters. Moreover, the Gaussian kernel function $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / \gamma^2)$ was only considered as it is often employed and yields great generalization performance. The penalty parameters in TSVM, WLTSVM, LSTSVM and KNN-LSTSVM are selected from set $\{2^i \mid i = -10, -9, \ldots, 9, 10\}$. The Gaussian kernel parameter γ is selected from the set $\{2^i \mid i = -15, -14, \ldots, 5\}$. The neighborhood size k in WLTSVM and KNN-LSTSVM is chosen from the set $\{2, 3, \ldots, 10\}$.

4.3 Synthetic datasets

To illustrate graphically advantage of our KNN-LSTSVM over LSTSVM, we conducted experiments on artificial

---

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

[2] The Cython is a superset of the Python programming language and generates efficient C code. More info at http://cython.org
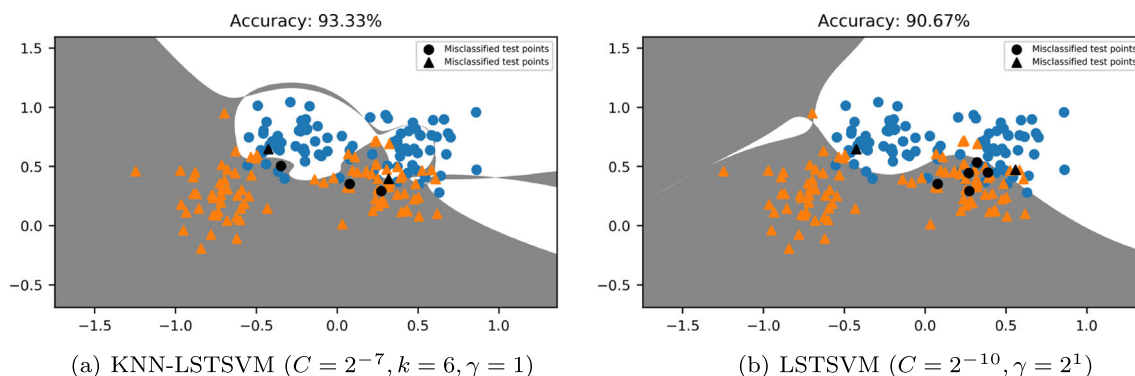
Fig. 4 The performance and decision boundary of KNN-LSTSVM and LSTSVM on Ripley's dataset with Gaussian kernel

Ripley's datset [32] which contains 250 samples and checkerboard dataset [11] of 1000 samples. 70% of samples were selected randomly for training both algorithms. Figures 4 and 5 depict performance of KNN-LSTSVM and LSTSVM on Ripey's and checkerboard dataset, respectively. It can be seen that KNN-LSTSVM obtains better separating hypersurface. This indicates that our proposed method significantly improves the classification performance of LSTSVM.

4.4 Experimental results and discussion

Table 2 shows the averages and standard deviation of the test accuracies (in %) for the TSVM, WLTSVM, LSTSVM and KNN-LSTSVM on fourteen UCI benchmark datasets. In addition, the training time of each classifier is demonstrated (It should be noted that the training time of WLTSVM and KNN-LSTSVM include KNN finding). The optimal parameters of four algorithms used in the experiments are also shown in Table 2. From the experimental results, we can draw the following conclusions:

1. From the perspective of classification accuracy, our proposed method (KNN-LSTSVM) outperforms the other three algorithms. This result validates that the

necessity of introducing similarity information within samples into the objective function which improves the accuracy of classifier.

2. In terms of computational time, LSTSVM is the fastest algorithm, because it solves two systems of linear equations. Even though two systems of linear equations are solved in our KNN-LSTSVM, KNN finding increases computing time. In comparison to WLTSVM, our proposed method costs shorter computational time as it inherits the advantage of LSTSVM which is solving a pair of linear equations rather than QPPs.

3. In order to show the effect of parameter k on the accuracy of KNN-LSTSVM, an experiment conducted on the Australian and Hepatits datasets. The value of k ranges from 2 to 30, and the step is 2. Figure 6 indicates the importance of selecting parameter k. The maximum accuracy appears when k is equal to 12. This implies that $k = 12$ is an optimal choice for the Australian dataset. For Hepatits dataset, the classification accuracy improves significantly as the value of k increases. Therefore, the choice of optimal parameter k is prominent.

4. In comparison with new non-parallel classifiers (i.e. RELS-TSVM [37] and Iv-TBSVM [41]), our KNN-LSTSVM has better classification ability as shown
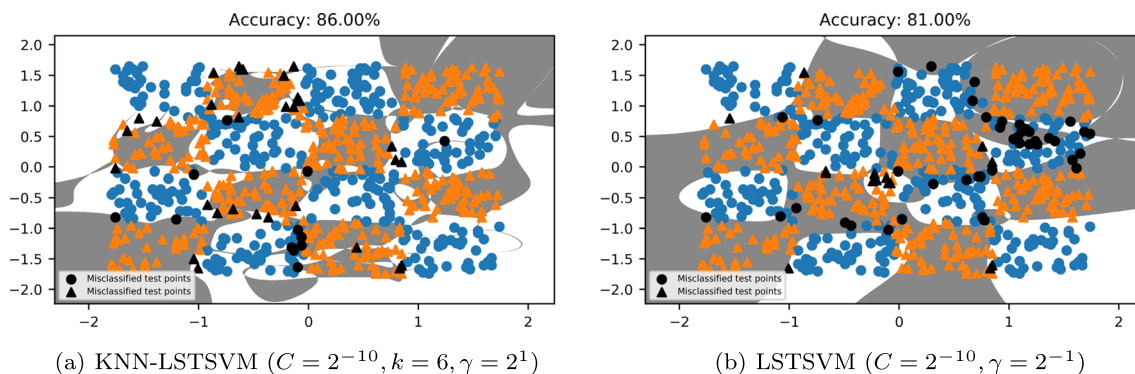

Fig. 5 The performance and decision boundary of KNN-LSTSVM and LSTSVM on checkerboard dataset with Gaussian kernel

Table 2 Comparison of accuracy and training time for TSVM, WLTSVM, LSTSVM and KNN-LSTSVM

Datasets	TSVM		WLTSVM		LSTSVM		KNN-LSTSVM	
$(m \times n)$	Accuracy(%) (C_1, C_2, γ)	Time(s)	Accuracy(%) (C, γ, k)	Time(s)	Accuracy(%) (C_1, C_2, γ)	Time(s)	Accuracy(%) (C, γ, k)	Time(s)
Austrailian	**87.97±2.75**	0.070	87.25±3.65	0.264	87.68±3.79	0.058	87.39±3.31	0.230
(690×14)	$(2^{-4}, 2^{-4}, 2^{-6})$		$(2^{-1}, 2^{-14}, 6)$		$(2^{-7}, 2^{-5}, 2^{-6})$		$(2^{3}, 2^{-13}, 7)$	
Bupa-Liver	73.62±4.77	0.082	74.82±3.84	0.193	74.51±6.89	0.007	**75.96±5.40**	0.041
(345×6)	$(2^{-3}, 2^{-3}, 2^{-6})$		$(2^{2}, 2^{-7}, 4)$		$(2^{5}, 2^{4}, 2^{-6})$		$(2^{10}, 2^{-6}, 7)$	
Cleveland	84.86±3.76	0.037	84.48±6.29	0.082	85.49±4.92	0.005	**85.51±6.41**	0.026
(303×13)	$(2^{0}, 2^{-1}, 2^{-11})$		$(2^{3}, 2^{-9}, 7)$		$(2^{2}, 2^{1}, 2^{-11})$		$(2^{-2}, 2^{-15}, 5)$	
Haber-Man	76.14±3.60	0.082	75.80±5.22	0.138	76.73±7.83	0.005	**76.81±5.82**	0.031
(306×3)	$(2^{1}, 2^{-1}, 2^{-9})$		$(2^{3}, 2^{-4}, 2)$		$(2^{9}, 2^{9}, 2^{-7})$		$(2^{-10}, 2^{-9}, 9)$	
Heart-Statlog	**85.56±5.84**	0.041	85.19±5.49	0.032	85.19±6.20	0.004	85.19±5.74	0.022
(270×13)	$(2^{1}, 2^{0}, 2^{-11})$		$(2^{1}, 2^{-13}, 6)$		$(2^{0}, 2^{-1}, 2^{-12})$		$(2^{-1}, 2^{-14}, 7)$	
Hepatits	85.79±8.65	0.003	86.50±8.79	0.040	**87.79±6.57**	0.001	87.13±7.59	0.006
(155×19)	$(2^{-4}, 2^{-2}, 2^{-8})$		$(2^{8}, 2^{-3}, 3)$		$(2^{3}, 2^{5}, 2^{-11})$		$(2^{-1}, 2^{-5}, 7)$	
Ionsphere	**92.59±3.19**	0.028	92.04±5.50	0.126	91.74±4.32	0.006	**92.59±4.47**	0.040
(351×34)	$(2^{-1}, 2^{-3}, 2^{-5})$		$(2^{1}, 2^{1}, 3)$		$(2^{6}, 2^{1}, 2^{-5})$		$(2^{1}, 2^{-5}, 5)$	
Monk3	98.37±1.26	0.150	98.38±1.69	0.494	98.55±1.36	0.032	**98.56±1.34**	0.126
(554×6)	$(2^{-4}, 2^{2}, 2^{-3})$		$(2^{3}, 2^{-6}, 2)$		$(2^{-6}, 2^{-3}, 2^{-3})$		$(2^{0}, 2^{-3}, 5)$	
Pima-Indian	77.87±4.73	0.147	77.86±3.49	0.566	77.61±5.89	0.073	**78.01±3.64**	0.339
(768×8)	$(2^{1}, 2^{1}, 2^{-3})$		$(2^{4}, 2^{-5}, 2)$		$(2^{-1}, 2^{-1}, 2^{-4})$		$(2^{-1}, 2^{-4}, 10)$	
Sonar	86.14±8.35	0.012	**87.50±3.17**	0.033	85.55±8.31	0.002	87.48±6.65	0.011
(208×60)	$(2^{-5}, 2^{-1}, 2^{-3})$		$(2^{5}, 2^{-4}, 7)$		$(2^{-10}, 2^{3}, 2^{-3})$		$(2^{-4}, 2^{-6}, 4)$	
Titanic	81.94±3.23	0.225	81.49±4.70	0.692	**82.38±4.63**	0.108	82.27±3.80	0.486
(891×7)	$(2^{-4}, 2^{-5}, 2^{-3})$		$(2^{0}, 2^{-6}, 7)$		$(2^{1}, 2^{1}, 2^{-5})$		$(2^{8}, 2^{-5}, 10)$	
Votes	96.78±2.11	0.145	97.01±1.79	0.098	**97.02±2.89**	0.012	97.01±3.11	0.064
(435×16)	$(2^{-6}, 2^{-3}, 2^{-6})$		$(2^{7}, 2^{-13}, 7)$		$(2^{-6}, 2^{-3}, 2^{-9})$		$(2^{6}, 2^{-9}, 3)$	
Wdbc	**98.42±1.23**	0.003	97.54±2.51	0.206	98.07±2.54	0.034	97.72±1.12	0.154
(569×30)	$(2^{-3}, 2^{-1}, 2^{-8})$		$(2^{-2}, 2^{-5}, 6)$		$(2^{-4}, 2^{-2}, 2^{-8})$		$(2^{-5}, 2^{-7}, 5)$	
Wpbc	82.39±9.20	0.006	80.84±8.94	0.019	81.32±9.01	0.002	**82.76±5.48**	0.010
(198×33)	$(2^{-4}, 2^{-5}, 2^{-6})$		$(2^{1}, 2^{-10}, 2)$		$(2^{-2}, 2^{-4}, 2^{-7})$		$(2^{-2}, 2^{-7}, 6)$	
Mean accuracy	86.31		86.19		86.40		**86.74**	

Bold value denotes the best result
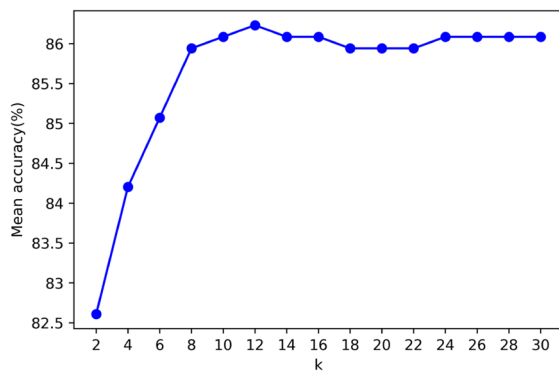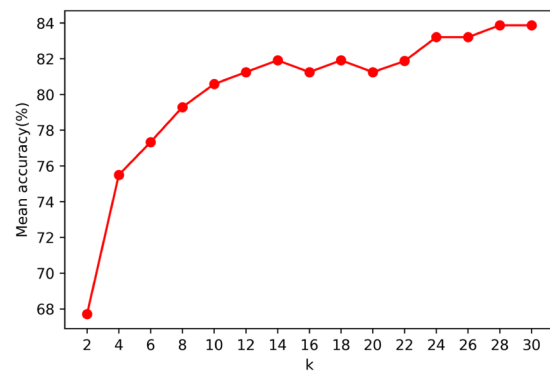
(a) Australian dataset. (b) Hepatits dataset.

Fig. 6 Changes of accuracy with the growth of k

Table 3 The comparison of accuracy for KNN-LSTSVM, RELS-TSVM and Iv-TBSVM in the case of Guassian kernel

Datasets	RELS-TSVM [37]		Iv-TBSVM [41]		KNN-LSTSVM	
	Accuracy(%)	Rank	Accuracy(%)	Rank	Accuracy(%)	Rank
Austrailian	86.00	2	84.79±6.26	3	**87.39±3.31**	**1**
Bupa-Liver	69.23	3	74.32±5.04	2	**75.96±5.40**	**1**
Haber-Man	77.35	2	**83.33±8.59**	**1**	76.81±5.40	3
Heart-Statlog	**85.71**	**1**	84.81±1.17	2	85.19±5.84	2
Ionsphere	**96.19**	**1**	95.61±4.11	2	92.59±4.47	3
Pima-Indian	**80.52**	**1**	77.56±4.13	3	78.01±3.64	2
Votes	96.90	2	96.51±2.26	3	**97.01±3.11**	**1**
Wdbc	85.51	3	**99.28±1.02**	**1**	97.72±1.12	2
Wpbc	80.70	2	79.90±4.13	3	**82.76±5.48**	**1**
Average rank	1.88		2.22		**1.77**	

Bold value denotes the best results

in Table 3. It obtains the lowest rank among new alogorithms. The results of RLES-TSVM and Iv-TBSVM were taken from [37] and [41], respectively. It should be noted that Iv-TBSVM solves two QPPs as opposed to solving a pair of linear equations in KNN-LSTSVM and RELS-TSVM. As a result, its learning speed is slower than least squares algorithms.

4.5 Statistical test

To further analyze the performance of four algorithms on fourteen UCI datasets as it was suggested in [7]. The Friedman test was used with corresponding post hoc tests which is proved to be a simple, nonparametric and safe. For this, the average ranks of four algorithms on accuracy for all datasets are calculated and listed in Table 4. Under the

Table 4 Average rank on classification accuracy of four algorithms

Datasets	TSVM	WLTSVM	LSTSVM	KNN-LSTSVM
Austrailian	1	4	2	3
Bupa-Liver	4	2	3	1
Cleveland	3	4	2	1
Haber-Man	3	4	2	1
Heart-Statlog	1	3	3	3
Hepatits	4	3	1	2
Ionsphere	1.5	3	4	1.5
Monk3	4	3	2	1
Pima-Indian	2	3	4	1
Sonar	3	1	4	2
Titanic	3	4	1	2
Votes	4	2.5	1	2.5
Wdbc	1	4	2	3
Wpbc	2	4	3	1
Average rank	2.61	3.18	2.43	**1.79**

Bold value denotes the best results

null hypothesis that all the algorithms are equivalent, one can compute the Friedman test according to (45):

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right], \qquad (45)$$

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

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \qquad (46)$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. We can obtain $\chi_F^2 = 8.293$ and $F_F = 3.198$ according to (45) and (46). Where F_F is distributed according to F-distribution with (3, 39) degrees of freedom. The critical value of $F(3, 39)$ is 1.42 for the level of significance $\alpha = 0.25$, similarly, it is 2.23 for $\alpha = 0.1$ and 2.84 for $\alpha = 0.05$. Since the value of F_F is larger than the critical value, there is significant difference between the four algorithms. Table 4 also illustrate that our KNN-LSTSVM is more valid than other three algorithms, because the average rank of KNN-LSTSVM is the lowest among other algorithms (Table 4).

4.6 Experiments with NDC datasets

In this subsection, experiments were conducted on large datasets to compare computing time of KNN-LSTSVM with other three algorithms as the number of datapoints increase. Large datasets were generated using NDC Data Generator [24]. Table 5 shows the description of NDC datasets. For experiments with these datasets, the penalty parameters of all algorithms were fixed to be one (i.e, C=1). The Gaussian kernel (RBF) was used with $\gamma = 2^{-15}$ for non-linear case. The neighborhood size k is also 5 for all

Table 5 The description of NDC datasets

Datasets	#Training data	#Test data	#Features
NDC-500	500	50	32
NDC-700	700	70	32
NDC-900	900	90	32
NDC-1K	1000	100	32
NDC-2K	2000	200	32
NDC-3K	3000	300	32
NDC-4K	4000	400	32
NDC-5K	5000	500	32
NDC-10K	10000	1000	32
NDC-25K	25000	2500	32
NDC-50K	50000	5000	32

Fig. 7 The changes of time with the growth of k on the NDC-10K dataset

datasets. Table 6 shows the comparison of computing time for all four algorithms on NDC datasets with both linear and Gaussian kernel.

KNN-LSTSVM performed several orders of magnitude faster than WLTSVM on all datasets, because KNN-LSTSVM does not require any external optimizer whereas WLTSVM is implemented with clipDCD algorithm. While KNN-LSTSVM is fast, it is not as fast as LSTSVM which is evident from Table 6. LSTSVM obviously requires solving two systems of linear equations whereas KNN-LSTSVM requires KNN-finding plus solving two systems of linear equations. For non-linear test, a rectangular kernel [22] was employed using 10% of total datapoints. Results with reduced kernel indicate that LSTSVM and KNN-LSTSVM are much faster than TSVM and WLTSVM, because even
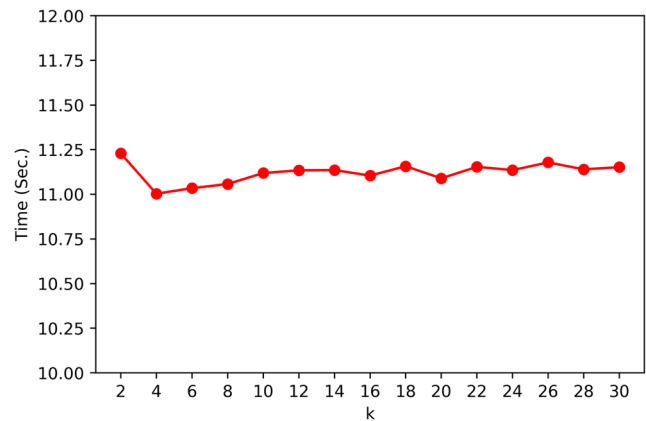
with reduced kernel ($m \times \bar{m}$), TSVM and WLTSVM still require solving two QPPs of size m_1 and m_2.

In order to show the influence of parameter k on the computing time of KNN-LSTSVM, an experiment was conducted on the relatively large dataset, NDC-10K. As shown in the Fig. 7, the value of k does not influence the computing time for KNN-LSTSVM. In summary, results on NDC datasets reveal that our KNN-LSTSVM is suitable for medium and large problems.

4.7 Application in human action recognition

To further investigate the efficiency and robustness of our KNN-LSTSVM, we apply it into human action recognition which is one of the active research area in the field of

Table 6 The comparison of computing time for four algorithms on NDC datasets

Datasets	Kernel	TSVM Time (s)		WLTSVM Time (s)		LSTSVM Time (s)		KNN-LSTSVM Time (s)	
		Linear	RBF	Linear	RBF	Linear	RBF	Linear	RBF
NDC-500		0.222	0.324	0.583	0.838	0.004	0.034	0.031	0.399
NDC-700		0.45	0.744	1.115	1.741	0.004	0.064	0.053	0.757
NDC-900		0.83	1.111	1.884	2.761	0.004	0.223	0.084	1.362
NDC-1K		0.976	1.599	2.315	3.499	0.004	0.266	0.1	1.657
NDC-2K		6.606	12.746	16.297	22.474	0.004	1.055	0.387	6.21
NDC-3K		25.387	41.908	73.451	70.213	0.004	2.588	0.904	14.041
NDC-4K		72.19	129.932	163.249	176.899	0.005	5.314	1.647	25.812
NDC-5K[b]		137.909	277.463	319.574	339.347	0.005	10.024	2.618	39.788
NDC-10K[b]		696.562	a	a	a	0.007	67.475	11.249	164.655
NDC-25K[b]		a	a	a	a	0.011	967.458	75.707	a
NDC-50K[b]		a	a	a	a	0.02	a	383.829	a

[a]We terminated the algorithm as computing time was very high

[b]A rectangular kernel $K(A, \bar{A})$ with $\bar{A}$ typically of size 10% of A was used

A. Mir, J. A. Nasiri

Fig. 8 Examples of sequences corresponding to different types of actions from KTH dataset

(a) Boxing

(b) Hand clapping

(c) Hand waving

(d) Jogging

(e) Running

(f) Walking

computer vision and pattern recognition. It has a wide range of applications in surveillance video, automatic video retrieval and human computer interaction [1]. The difficulty of human action recognition problems may originate from several challenges such as illumination changes, partial occlusions, and intra-class variations [26].

An action is a sequence of body movements that may involves several body parts concurrently [5]. From the perspective of computer vision, the goal of human action recognition is to correctly classify the videos into its action category [1]. Major components of human action recognition system include action representation and classification.

4.7.1 Action representation

Sparse space-time features have been frequently used in human action recognition research which are extracted from 3-dimensional space-time volumes to represent and classify actions [1]. Space-time interest points are detected using Harris3D operator [18]. The Histogram of Oriented Gradient (HoG) and Optical Flow (HOF) were employed for action representation [19].

After extracting space-time features, the Bag of Video Words (BoVW) technique was applied which treats videos as documents and visual features as words. This technique proved its robustness to location changes and to noise. The visual words are constructed by utilizing K-means clustering [21] as it is the most popular algorithm to construct visual dictionary. Since the total number of features is very large to use for clustering, a subset of 10^5 features were selected randomly. The number of clusters is set to 4000 which has shown empirically to produce good results. The resulted clusters are equivalent to the visual

words. Finally, the word frequency histogram are computed and used as video sequence representation.

4.7.2 KTH dataset

In Section 4.7, the experiments are carried out on KTH dataset which was introduced by Schuldt et al. [34] and is one of the most popular human activity dataset. It contains six types of human actions (boxing, hand clipping, hand waving, jogging, running and walking) performed by 25 people in four different scenarios including outdoor, indoor, changes in clothing and variations in scale. The dataset consists of 2391 sequences where the background is homogeneous in all cases. Figure 8 shows several examples frames corresponding to different types of action.

4.7.3 Results and discussion

Due to the limited number of persons in the KTH dataset, the leave-one-person-out is used where each run uses 24 persons for training and one person for testing. Then, the average of the results is calculated to give the final recognition rate.

For experiment with KTH dataset, our linear KNN-LSTSVM was extended to handle multi-class problems based on One-versus-All (OVA) strategy. For K-class classification problem, the linear OVA KNN-LSTSVM solves K-linear equations and determines K non-parallel hyper planes, one for each class [39]. Table 7 shows a comparison between our KNN-LSTSVM and other classifiers on KTH dataset. The result shows that our proposed method has the highest mean accuracy among other algorithms. This further validates the effectiveness of our KNN-LSTSVM in real-world application.

Springer

Table 7 Mean accuracy rates of different methods on KTH dataset

Method	Mean accuracy (%)	Time (s)
Standard SVM	91.8	18430
LSTSVM	92.33	60.29
KNN-LSTSVM	**93.54**	245.87

Bold value denotes the best results

5 Conclusion

Motivated by both LSTSVM and WLTSVM, we present a novel algorithm, i.e, the KNN-based least squares twin support vector machine. Similar to WLTSVM, our new algorithm takes full advantage of similarity information within each samples by KNN method before classification which improves prediction accuracy. Furthermore, the proposed method not only inherits the advantage of LSTSVM algorithm, which owns low computational time by solving linear equations, but also addresses the shortcoming of outlier sensitivity and noise tolerance in LSTSVM. The experimental results reveal that KNN-LSTSVM outperforms other three algorithms in terms of classification accuracy. The computational result on NDC datasets also indicates that KNN-LSTSVM is faster than WLTSVM and can handle large-scale classification problems. We also investigated the application of linear KNN-LSTSVM to human action recognition. The comparison of experimental results against linear LSTSVM show that linear KNN-LSTSVM has better classification ability on KTH dataset. The subject of our future work is to find effective methods which improves KNN-LSTSVM in terms of learning speed and memory consumption.

References

1. Aggarwal JK, Ryoo MS (2011) Human activity analysis: a review. ACM Comput Surv (CSUR) 43(3):16
2. Behnel S, Bradshaw R, Citro C, Dalcin L, Seljebotn DS, Smith K (2011) Cython: the best of both worlds. Comput Sci Eng 13(2):31–39
3. Bishop CM (2006) Pattern recognition and machine learning. Springer
4. Cai D, He X, Zhou K, Han J, Bao H (2007) Locality sensitive discriminant analysis. In: IJCAI, vol 2007, pp 1713–1726
5. Cheng G, Wan Y, Saudagar AN, Namuduri K, Buckles BP (2015) Advances in human action recognition: a survey. arXiv:150105964
6. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
7. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
8. Ding S, Yu J, Qi B, Huang H (2014) An overview on twin support vector machines. Artif Intell Rev 42(2):245–252
9. Ding S, Zhang N, Zhang X, Wu F (2017) Twin support vector machine: theory, algorithm and applications. Neural Comput and Applic 28(11):3119–3130
10. Golub GH, Van Loan CF (2012) Matrix computations, vol 3. JHU Press
11. Ho T, Kleinberg E (1996) Checkerboard dataset
12. Jayadeva KR, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29:5
13. Jayadeva KR, Chandra S (2017) Twin support vector machines: models extensions and applications. Springer Series on Computational Intelligence
14. Jones E, Oliphant T, Peterson P (2014) {SciPy}: open source scientific tools for {Python}
15. Khemchandani R, Saigal P, Chandra S (2016) Improvements on ν-twin support vector machine. Neural Netw 79:97–107
16. Khemchandani R, Saigal P, Chandra S (2017) Angle-based twin support vector machine. Ann Oper Res, 1–31
17. Kumar MA, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36(4):7535–7543
18. Laptev I (2005) On space-time interest points. Int J Comput Vis 64(2-3):107–123
19. Laptev I, Marszalek M, Schmid C, Rozenfeld B (2008) Learning realistic human actions from movies. In: IEEE Conference on computer vision and pattern recognition, 2008. CVPR 2008.IEEE, pp 1–8
20. Lee LH, Wan CH, Rajkumar R, Isa D (2012) An enhanced support vector machine classification framework by using euclidean distance function for text document categorization. Appl Intell 37(1):80–99
21. MacQueen J et al (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol 1, Oakland, pp 281-297
22. Mangasarian OL, Wild EW (2001) Proximal support vector machine classifiers. In: Proceedings KDD-2001, knowledge discovery and data mining. Citeseer
23. Mangasarian OL, Wild EW (2006) Multisurface proximal support vector machine classification via generalized eigenvalues. IEEE Trans Pattern Anal Mach Intell 28(1):69–74
24. Musicant D (1998) Ndc: normally distributed clustered datasets. Computer Sciences Department. University of Wisconsin, Madison
25. Nasiri JA, Naghibzadeh M, Yazdi HS, Naghibzadeh B (2009) Ecg arrhythmia classification with support vector machines and genetic algorithm. In: Third UKSim European symposium on computer modeling and simulation, 2009. EMS'09. IEEE, pp 187–192
26. Nasiri JA, Charkari NM, Mozafari K (2014) Energy-based model of least squares twin support vector machines for human action recognition. Signal Process 104:248–257
27. Nayak J, Naik B, Behera H (2015) A comprehensive survey on support vector machine in data mining tasks: applications & challenges. Int J Datab Theory Appl 8(1):169–186
28. Owusu E, Zhan Y, Mao QR (2014) An svm-adaboost facial expression recognition system. Appl Intell 40(3):536–545
29. Peng X, Chen D, Kong L (2014) A clipping dual coordinate descent algorithm for solving support vector machines. Knowl-Based Syst 71:266–278
30. Poppe R (2010) A survey on vision-based human action recognition. Image Vis Comput 28(6):976–990
31. Rastogi R, Saigal P, Chandra S (2018) Angle-based twin parametric-margin support vector machine for pattern classification. Knowl-Based Syst 139:64–77

32. Ripley BD (2007) Pattern recognition and neural networks. Cambridge University Press

33. Scherer S, Kane J, Gobl C, Schwenker F (2013) Investigating fuzzy-input fuzzy-output support vector machines for robust voice quality classification. Comput Speech Lang 27(1):263–287

34. Schuldt C, Laptev I, Caputo B (2004) Recognizing human actions: a local svm approach. In: Proceedings of the 17th International conference on pattern recognition, 2004. ICPR 2004, vol 3. IEEE, pp 32–36

35. Shao YH, Zhang CH, Wang XB, Deng NY (2011) Improvements on twin support vector machines. IEEE Trans Neural Netw 22(6):962–968

36. Smola AJ, Schölkopf B (1998) Learning with kernels. GMD-Forschungszentrum Informationstechnik

37. Tanveer M, Khan MA, Ho SS (2016) Robust energy-based least squares twin support vector machines. Appl Intell 45(1):174–186

38. Tian Y, Qi Z (2014) Review on: twin support vector machines. Ann Data Sci 1(2):253–277

39. Tomar D, Agarwal S (2015) A comparison on multi-class classification methods based on least squares twin support vector machine. Knowl-Based Syst 81:131–147

40. Svd Walt, Colbert SC, Varoquaux G (2011) The numpy array: a structure for efficient numerical computation. Computi Sci Eng 13(2):22–30

41. Wang H, Zhou Z, Xu Y (2018) An improved ν-twin bounded support vector machine. Appl Intell 48(4):1041–1053

42. 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