IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 November 2022, accepted 8 December 2022, date of publication 16 January 2023, date of current version 25 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3237641

THEORY

Deep Non-Parallel Hyperplane Support Vector Machine for Classification

FEIXIANG SUN AND XIJIONG XIE
School of Information Science and Engineering, Ningbo University, Ningbo 315211, China
Corresponding author: Xijiong Xie (xjxie11@gmail.com)

This work was supported by NSFC under Grant 61906101.

ABSTRACT In the last few decades, deep learning based on neural networks has become popular for the classification tasks, which combines feature extraction with the classification tasks and always achieves the satisfactory performance. Non-parallel hyperplane support vector machine (NPHSVM) aims at constructing two non-parallel hyperplanes to classify data and extracted features are always used to be input data for NPHSVM. As for NPHSVM, extracted features will greatly influence the performance of the model to some extent. Therefore, in this paper, we propose a novel DNHSVM for classification, which combines deep feature extraction with the generation of hyperplanes seamlessly. Each hyperplane is close to its own class and as far as possible to other classes, and deep features are friendly for classification and samples are easy to be classified. Experiments on UCI datasets show the effectiveness of our proposed method, which outperforms other compared state-of-the-art algorithms.

INDEX TERMS Deep learning, non-parallel hyperplane support vector machine, feature extraction.

I. INTRODUCTION

Support vector machine (SVM) [1] achieves relatively good performance in many applications, which aims at constructing a hyperplane that can classify samples with the structure risk minimization principle. In recent decades, several variants for SVM have been proposed, such as least squares SVM (LSSVM) [2], pinball SVM (Pin-SVM) [3], robust support vector classifiers (RSVC) [4] and so on.

However, SVM strictly requires two parallel hyperplanes, which will meet challenges when solving some learning tasks. Consequently, Mangasarian et al. [5] proposed generalized eigenvalue support vector machine (GEPSVM), which can construct two non-parallel hyperplanes by solving generalized eigenvalue problems. After that, Tian et al. presented non-parallel support vector machine for pattern classification [6]. The ϵ-insensitive loss function was introduced to two primal problems instead of the quadratic loss function. Li et al. [7] proposed robust L_1 norm non-parallel proximal support vector machine, which uses an iterative technique to solve a pair of L_1 norm optimal problems because L_1 norm is robust to outliers. Twin support vector machine (TSVM) [8] was presented by Jayadeva et al., which attempts to construct two non-parallel hyperplanes by solving two small quadratic programming problems (QPPs). In the last few years, many variants for TSVM have also been proposed. For example, Gao et al. proposed L_1 norm least squares twin support vector machine [9], which can automatically select relevant features. Qi et al. [10] presented structural twin support vector machine for classification, which can fully exploit the prior structural information to enhance the classification accuracy. Wang et al. proposed robust capped L_1 norm twin support vector machine [11] which is more robust to outliers, and they designed a simple and efficient algorithm to solve the resultant objective. Xie et al. [12] proposed multitask twin support vector machine, which learns multiple related tasks simultaneously and puts TSVM to multitask learning. They also proposed multitask centroid twin support vector machine [13], which overcomes the disadvantage that TSVM is sensitive to outliers. In addition, Xie et al. also extended TSVM to multi-view learning, such as general multi-view semi-supervised least squares support vector machines with multi-manifold regularization [14], multi-view twin support vector machines [15], multi-view support vector machines with the consensus and complementary information [16], multi-view semi-supervised least squares twin support vector

The associate editor coordinating the review of this manuscript and approving it for publication was Zahid Akhtar.

machines with manifold-preserving graph reduction [17] and so on. Shao et al. [18] proposed an efficient weighted Lagrangian twin support vector machine for imbalanced data classification. They introduced a graph-based under-sampling strategy to keep the proximal information, which is robust to outliers. In addition, the weight biases are embedded in Lagrangian TSVM formulations, which overcomes the bias phenomenon for imbalanced data classification. Li et al. [19] proposed generalized elastic net L_p norm non-parallel support vector machine. L_p norm is used to measure the distance of samples to each hyperplane and an appropriate p can be chosen to achieve the desired performance. Li et al. [20] proposed domain adaptive twin support machine learning using privileged information, and they also proposed an effective method to generate two non-parallel hyperplanes to minimize the distance between the source domain and target domain. Xie et al. [21] proposed Laplacian Lp norm least squares twin support vector machine, and the performance can be improved with the appopriate p value and Lp norm graph regularization term. Rezvani et al. [22] proposed intuitionistic fuzzy twin support vector machinesd for imbalanced data, which can easily deal with imbalanced datasets in the presence of noises and outliers. Chen et al. [23] proposed fuzzy support vector machine with graph for classifying imbalanced datasets. Firstly, they designed a graph-based fuzzy membership function to assess the importance of samples in the original feature space. Secondly, they proved that the function can mine discriminative information between samples in high-dimensional data. Thirdly, a method was provided to calculate the fuzzy membership function in the kernel space. Finally, the model analyzes samples of each class independently.

In recent years, SVM and TSVM have been extended to the field of deep learning. For example, Li et al. presented deep twin support vector machine [24], which regards the projection as a kind of location information and projects raw data into 4-dimensional space to get new data. However, it does not use deep neural network (DNN) to extract deep features. Deep learning using support vector machines [25] was proposed by Tang. In addition, there are also some applications for deep SVM. For example, Wu et al. [26] proposed deep two-view support vector machine for facial expression recognition. The visible and thermal facial images are viewed as two views. Two deep neural networks are trained for visible and thermal image data. In addition, Okwuashi et al. proposed deep support vector machine for hyperspectral image classification [27].

The kernel trick can be used to project the original data to higher dimension space, which can be used to handle linear nonseparable problem. Besides, neural network can also be used to extract features. In this paper, inspired by the success of non-parallel hyperplane support vector machine (NPHSVM) [28] and deep learning, we propose a novel deep non-parallel hyperplane support vector machine (DNHSVM) for classification.

The contributions of this paper are summarized as follows:

(1) We extend non-parallel hyperplane support vector machine to deep learning field, which can combine feature extraction with the task of classification seamlessly.

(2) The parameters of neural networks and hyperplanes can be optimized simultaneously by the algorithm of stochastic gradient descent (SGD), and extracted deep features are friendly for classification.

(3) Experiments on UCI datasets demonstrate the effectiveness of our proposed method. Our proposed method outperforms other state-of-the-art compared methods.

The remainder of this paper is organized as follows: Section II gives a brief introduction of SVM, TSVM, NPHSVM and deep support vector machine (DSVM). Section III elaborates our proposed DNHSVM and its optimization procedure. Section IV shows experiments for our proposed method and compared methods on benchmark datasets. Finally, in section V, we make the conclusion.

II. RELATED WORK

Notations are summarized here throughout this paper. Matrices are written in uppercase. As for matrix $A \in \mathbb{R}^{n\times d}$, A_i denotes the *ith* row of matrix A and the transpose of matrix A is denoted as $A^\top$. A^{-1} is the inverse matrix of A, I is an identity matrix. The vectors and scalar are written in lowercase.

A. SUPPORT VECTOR MACHINE

Suppose we are given n training data as matrix $A \in \mathbb{R}^{n\times d}$, where d is the number of dimension of samples. The *ith* row of matrix A represents the *ith* sample and label for the *ith* training data is $y_i, y_i \in \{-1, 1\}$. $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are normal vector and bias, respectively. In order to obtain hyperplane with the structure risk minimization principle, the following constraint should be satisfied

$$y_i(A_i w + b) \geq 1, \quad i = 1, 2, \cdots, n. \tag{1}$$

The hyperplane is denoted as $w^\top x + b = 0$ and the margin between two hyperplanes is $\frac{2}{\|w\|}$. The standard SVM is given by solving the following optimization problem

$$\begin{aligned} &\min_{w,b} \frac{1}{2} w^\top w \\ &\text{s.t.} \quad y_i(A_i w + b) \geq 1, \quad i = 1, 2, \cdots, n. \end{aligned} \tag{2}$$

The parameters of hyperplane can be obtained by solving a complex QPP, then the decision function can be written as

$$f(x) = \text{sign}(w^\top x + b). \tag{3}$$

The test sample $x \in \mathbb{R}^d$ can be assigned to positive class "+1" or negative class "−1" according to which side of hyperplane it is on.

B. TWIN SUPPORT VECTOR MACHINE

TSVM attempts to construct two non-parallel hyperplanes by solve a pair of small QPPs and each hyperplane is close to its

own class and keeps away from the other class. Positive and negative samples are denoted as $A \in \mathbb{R}^{n_1 \times d}$ and $B \in \mathbb{R}^{n_2 \times d}$, respectively, where n_1 and n_2 are the numbers of positive and negative samples respectively and d is the dimension of samples. Two hyperplanes can be given as:

$$x^\top w_1 + b_1 = 0 \quad \text{and} \quad x^\top w_2 + b_2 = 0, \tag{4}$$

where w_1 and w_2 are normal vectors of hyperplanes, b_1 and b_2 are bias terms. The primal problems of TSVM can be formulated as

$$\begin{aligned} \min_{w_1,b_1,\xi} \quad & \frac{1}{2}\|Aw_1 + e_1 b_1\|^2 + c_1 e_2^\top \xi \\ \text{s.t.} \quad & -(Bw_1 + e_2 b_1) + \xi \geq e_2, \xi \geq 0, \end{aligned} \tag{5}$$

$$\begin{aligned} \min_{w_2,b_2,\eta} \quad & \frac{1}{2}\|Bw_2 + e_2 b_2\|^2 + c_2 e_1^\top \eta \\ \text{s.t.} \quad & (Aw_2 + e_1 b_2) + \eta \geq e_1, \eta \geq 0, \end{aligned} \tag{6}$$

where ξ and η are slack variables, c_1 and c_2 are hyperparameters which need to be tuned, e_1 and e_2 are vectors whose all elements are ones with the appropriate dimension. After introducing the Lagrangian multipliers α and β, Eq. (5) can be written as

$$\begin{aligned} L(w_1, b_1, \xi, \alpha, \beta) = & \frac{1}{2}(Aw_1 + e_1 b_1)^\top (Aw_1 + e_1 b_1) \\ & + c_1 e_2^\top \xi - \alpha^\top - (Bw_1 + e_2 b_1) \\ & + \xi - e_2 - \beta^\top \xi, \end{aligned} \tag{7}$$

with KKT conditions, the following equations can be obtained

$$A^\top (Aw_1 + e_1 b_1) + B^\top \alpha = 0, \tag{8}$$

$$e_1^\top (Aw_1 + e_1 b_1) + e_2^\top \alpha = 0, \tag{9}$$

$$c_1 e_2 - \alpha - \beta = 0, \tag{10}$$

$$-(Bw_1 + e_2 b_1) + \xi \geq e_2, \quad \xi \geq 0, \tag{11}$$

$$\alpha^\top (-(Bw_1 + e_2 b_1) + \xi - e_2) = 0, \beta^\top \xi = 0, \alpha \geq 0, \quad \beta \geq 0. \tag{12}$$

From the KKT condition and Eq. (5), Eq. (13) can be obtained. Using the similar method, Eq. (14) can also be obtained.

$$\begin{aligned} \max_{\alpha} \ & \alpha^\top e_2 - \frac{1}{2}\alpha^\top H (G^\top G)^{-1} H^\top \alpha \\ \text{s.t. } & 0 \leq \alpha \leq c_1, \end{aligned} \tag{13}$$

$$\begin{aligned} \max_{\beta} \ & \beta^\top e_1 - \frac{1}{2}\beta^\top G (H^\top H)^{-1} G^\top \beta \\ \text{s.t.} \quad & 0 \leq \beta \leq c_2, \end{aligned} \tag{14}$$

where $G = [A \ e_1], H = [B \ e_2]$, vectors α and β are Lagrangian multipliers. After solving QPPs, parameters of two hyperplanes can be formulated as:

$$u_1 = -(G^\top G + \delta I)^{-1} H^\top \alpha, \tag{15}$$

$$u_2 = (H^\top H + \delta I)^{-1} G^\top \beta, \tag{16}$$

where $u_1 = [w_1^\top \ b_1]^\top, u_2 = [w_2^\top \ b_2]^\top$, I is an identity matrix of the appropriate dimension, δI is a regularization term to solve challenge that the inverse of $G^\top G$ or $H^\top H$ is difficult to obtain. After we obtain parameters of two hyperplanes, a new sample $x \in \mathbb{R}^d$ can be assigned to positive class "+1" or negative class "−1" depending on which hyperplane it is closer to, i.e.

$$f(x) = \text{sign}\left(\frac{|w_1^\top x + b_1|}{\|w_1\|} - \frac{|w_2^\top x + b_2|}{\|w_2\|}\right). \tag{17}$$

C. NONPARALLEL HYPERPLANE SUPPORT VECTOR MACHINE

Nonparallel hyperplane support vector machine aims at constructing two hyperplanes $f_1(x) = w_1^\top x + b_1$ and $f_2(x) = w_2^\top x + b_2$. Each hyperplane is close to its own class and far away from the other class. It maximizes the differences of $(X_1 w_1 + e_1 b_1) - (X_1 w_2 + e_1 b_2)$ and $(X_2 w_2 + e_2 b_2) - (X_2 w_1 + e_2 b_1)$, and the primal for NPHSVM is as follows

$$\begin{aligned} \min_{w_1,b_1,w_2,b_2,\xi_1,\xi_2} \ & \frac{1}{2}(\|w_1\|^2 + b_1^2 + \|w_2\|^2 + b_2^2) \\ & + \frac{C_1}{2}(\|X_1 w_1 + e_1 b_1\|^2 + \|X_2 w_2 + e_2 b_2\|^2) \\ & + C_2(e_1^\top \xi_1 + e_2^\top \xi_2) \\ \text{s.t. } & X_1 w_1 + e_1 b_1 - X_1 w_2 - e_1 b_2 \geq e_1 - \xi_1, \\ & X_2 w_2 + e_2 b_2 - X_2 w_1 - e_2 b_1 \geq e_2 - \xi_2, \\ & \xi_1 \geq 0, \xi_2 \geq 0, \end{aligned} \tag{18}$$

where C_1 and C_2 are regularization parameters. There are three terms in equation (18), and the first is the regularization term. The second is the sum of squared distances from the two hyperplanes to samples in the corresponding classes which keeps the hyperplane close to its own class. The third is the sum of error variables corresponding to constraints. With the Lagrangian and Karush-Kuhn-Tucker (KKT) conditions, the dual to the problem can be obtained

$$\begin{aligned} \max_{\alpha} \ & -\frac{1}{2}\alpha^\top \hat{X}^\top [(I + C_1 \hat{X_1}^\top \hat{X_1})^{-1} \\ & + (I + C_1 \hat{X_2}^\top \hat{X_2})^{-1}] \hat{X} \alpha \\ & + e^\top \alpha \\ \text{s.t.} \quad & 0 \leq \alpha \leq C_2 e, \end{aligned} \tag{19}$$

where $\hat{X_1} = [X_1, e_1], \hat{X_2} = [X_2, e_2], \hat{X} = [\hat{X_1}, -\hat{X_2}], e = [e_1, e_2]$ and $\alpha = [\alpha_1, \alpha_2]$. The solutions for w_1, b_1, w_2, b_2 in equation (18) can be obtained from the solution of α. A new sample $x \in \mathbb{R}^m$ can be assigned to class "+1" or "−1" depending on which of the two hyperplanes it is close to.

D. DEEP SUPPORT VECTOR MACHINE

Deep support vector machine [25] extends SVM to the deep learning field. The top layer of neural network can be known as parameters of hyperplane which is used to classify samples. DSVM combines features extraction with the construction of hyperplane. The parameters of hyperplane can be optimized by SGD method and loss function for DSVM can

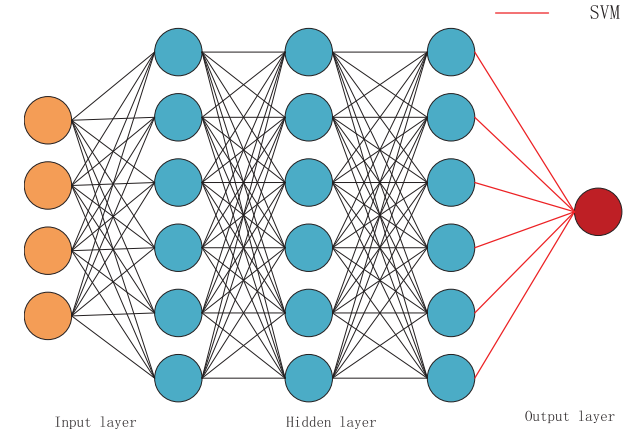

FIGURE 1. The model of DSVM.

be written as

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{n}\max(0, 1 - y_i(w^\top f^M(x_i) + b)), \quad (20)$$

where w is a normal vector for hyperplane and b is a bias term of DSVM, c is a hyperparameter that needs to be tuned, $f^M(x_i) = g(f^{M-1}(x_i)(w^M)^\top + b^M)$ is extracted features through M-layer neural network, $g(\cdot)$ is an activation function such as Relu or Sigmoid, w^M and b^M are weight and bias for Mth-layer of neural network. The model for DSVM is shown in Fig. 1. After neural network is trained, test samples can be assigned to corresponding labels according to which side of hyperplane they are on.

III. OUR PROPOSED DEEP NON-PARALLEL HYPERPLANE SUPPORT VECTOR MACHINE

A. BINARY CLASSIFICATION FOR DNHSVM

In this section, we elaborate our proposed deep non-parallel hyperplane support vector machine. The loss function for our DNHSVM can be defined as:

$$\begin{aligned}\mathcal{L} = {} & \lambda_1(\|f^M(A)w_1 + e_1b_1\|^2 + \|f^M(B)w_2 + e_2b_2\|^2) \\ & + \lambda_2(\|f^M(A)w_1 + e_1b_1 - f^M(A)w_2 - e_1b_2 - e_1\|^2 \\ & + \|f^M(B)w_2 + e_2b_2 - f^M(B)w_1 - e_2b_1 - e_2\|^2) \\ & + \frac{1}{2}(\|w_1\|^2 + \|w_2\|^2), \end{aligned} \quad (21)$$

where $f^M(A)$ and $f^M(B)$ are extracted features through M-layer neural network for positive and negative samples, respectively, w_1 and w_2 are normal vectors of hyperplanes, b_1 and b_2 are bias terms, e_1 and e_2 are vectors whose all elements are ones with the appropriate dimension.

To explain the principle of our DNHSVM, we provide the following analyses.

(1) The first term indicates the squared distance from samples to their own hyperplanes. The smaller the value is, the closer they are to their own hyperplanes.

(2) The second term shows that the difference between the distance from samples to their own hyperplane and the other hyperplane should be at least one, which can be used to classify two classes effectively.

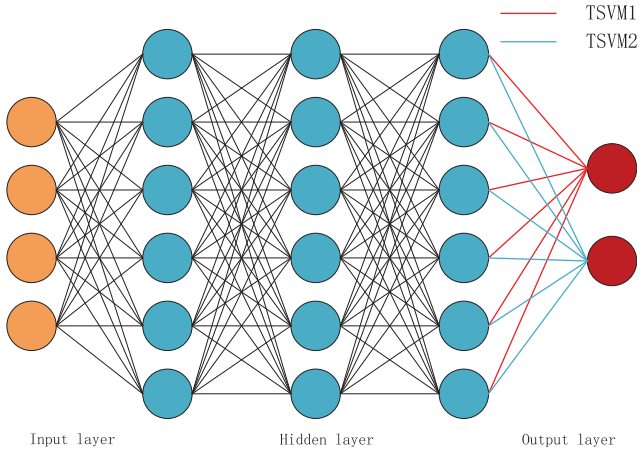

FIGURE 2. The proposed model of deep non-parallel hyperplane support vector machine.

(3) The third term $\frac{1}{2}(\|w_1\|^2 + \|w_2\|^2)$ is a regularization term which can be used to avoid over-fitting.

The model is shown in Fig. 2 and the red and blue lines are parameters of two non-parallel hyperplanes we seek.

We jointly optimize parameters of two hyperplanes $\{w_1, b_1, w_2, b_2\}$ and DNN parameter θ using SGD method. The gradients of $\mathcal{L}$ with respect to w_1, b_1, w_2, b_2 are shown as:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial w_1} = {} & 2\lambda_1 f^M(A)^\top S_1 + 2\lambda_2 f^M(A)^\top(S_1 - S_3 - e_1) \\ & - 2\lambda_2 f^M(B)^\top(S_2 - S_4 - e_2) + w_1, \end{aligned} \quad (22)$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial b_1} = {} & 2\lambda_1 e_1^\top S_1 + 2\lambda_2(e_1^\top(S_1 - S_3 - e_1) \\ & - e_2^\top(S_2 - S_4 - e_2)), \end{aligned} \quad (23)$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial w_2} = {} & 2\lambda_1 f^M(B)^\top S_2 - 2\lambda_2 f^M(A)^\top(S_1 - S_3 - e_1) \\ & + 2\lambda_2 f^M(B)^\top(S_2 - S_4 - e_2) + w_2, \end{aligned} \quad (24)$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial b_2} = {} & 2\lambda_1 e_2^\top S_2 - 2\lambda_2(e_1^\top(S_1 - S_3 - e_1) \\ & - e_2^\top(S_2 - S_4 - e_2)), \end{aligned} \quad (25)$$

where $S_1 = f^M(A)w_1 + e_1b_1$, $S_2 = f^M(B)w_2 + e_2b_2$, $S_3 = f^M(A)w_2 + e_1b_2$ and $S_4 = f^M(B)w_1 + e_2b_1$. The learning rate for neural network is denoted as α, then w_1, b_1, w_2, b_2 can be updated as follows:

$$w_1 = w_1 - \alpha\frac{\partial\mathcal{L}}{\partial w_1}, \quad b_1 = b_1 - \alpha\frac{\partial\mathcal{L}}{\partial b_1}, \quad (26)$$

$$w_2 = w_2 - \alpha\frac{\partial\mathcal{L}}{\partial w_2}, \quad b_2 = b_2 - \alpha\frac{\partial\mathcal{L}}{\partial b_2}. \quad (27)$$

The backward error will be passed to the previous layers. For M-layer, the backward error for the ith positive and negative samples is denoted as $\delta_{A_i}^M$ and $\delta_{B_i}^M$, respectively, i.e.

$$\begin{aligned}\delta_{A_i}^M = {} & \frac{\partial\mathcal{L}}{\partial f^M(A_i)} \cdot \frac{\partial f^M(A_i)}{\partial z_{A_i}^M} \\ = {} & (2\lambda_1(f^M(A_i)w_1 + b_1)w_1 + 2\lambda_2(f^M(A_i)w_1 + b_1 \end{aligned}$$

$$- f^M(A_i)w_2 - b_2 - 1)(w_1 - w_2)) \odot g'(z^M_{A_i}), \quad (28)$$

$$\delta^M_{B_i} = \frac{\partial \mathcal{L}}{\partial f^M(B_i)} \cdot \frac{\partial f^M(B_i)}{\partial z^M_{B_i}}$$
$$= (2\lambda_1(f^M(B_i)w_2 + b_2)w_2 + 2\lambda_2(f^M(B_i)w_2 + b_2$$
$$- f^M(B_i)w_1 - b_1 - 1)(w_2 - w_1)) \odot g'(z^M_{B_i}), \quad (29)$$

where $z^M_{A_i}$ and $z^M_{B_i}$ are positive and negative input data of M-layer neural network. $f^M(A_i)$ and $f^M(B_i)$ are the *ith* positive and negative samples for output data of M-layer, respectively, $\odot$ is element-wise multiplication, $g'(\cdot)$ is the derivative of activation function $g(\cdot)$. Then we can obtain the definition of $\delta_i^{(m)}$ as follows

$$\delta_i^{(m)} = \begin{cases} \left(W^{(m+1)}\delta_i^{(m+1)}\right) \odot g'\left(z^{(m)}_{A_i/B_i}\right) & m \neq M \\ \left(\delta^M_{A_i/B_i}\right) \odot g'\left(z^{(M)}_{A_i/B_i}\right) & m = M. \end{cases} \quad (30)$$

According to the definition of $f_i^{(m)}$ and back-propagation algorithm, subgradient with respect to W^m and b^m can be formulated as

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial W^{(m)}} = \delta_i^{(m)} f_i^{(m-1)T} \\ \dfrac{\partial \mathcal{L}}{\partial b^{(m)}} = \delta_i^{(m)}, \end{cases} \quad (31)$$

where $f_i^{(m-1)}$ represents the *ith* positive or negative sample output data of $(m-1)$-layer. Based on Eq. (30) and Eq. (31) with SGD algorithm, $W^{(m)}$ and $b^{(m)}$ can be updated as follows:

$$\begin{cases} W^{(m)} = W^{(m)} - \alpha \dfrac{\partial \mathcal{L}}{\partial W^{(m)}} \\ b^{(m)} = b^{(m)} - \alpha \dfrac{\partial \mathcal{L}}{\partial b^{(m)}}. \end{cases} \quad (32)$$

Before the training, parameters of deep neural network are initialized randomly. During the testing, decision function of our proposed model is shown in Eq. (33)

$$f(x) = \text{sign}\left(\frac{|w_1^\top f^M(x) + b_1|}{\|w_1\|} - \frac{|w_2^\top f^M(x) + b_2|}{\|w_2\|}\right), \quad (33)$$

where $f^M(x)$ is extracted feature through M-layer neural network. The algorithm of our proposed method is described in Algorithm 1.

B. MULTICLASS CLASSIFICATION FOR DNHSVM

An easy way to extend binary non-parallel support vector machine to classification is using $one - vs - one$ approach. As for K class problems, $\frac{K(K-1)}{2}$ DNHSVM models will be trained independently and the vote strategy is used to obtain the final results.

Algorithm 1 Deep Non-Parallel Support Vector Machine

1: **Input** : The positive samples A and negative samples B, hyperparameter λ_1 and λ_2 (selected by the strategy of cross-validation), the learning rate α; maximum epochs E, mini-batch m_b, number of batch M;
2: **Output** : The label of test samples is according to Eq. (33);
3: **Initial** : Initialize the parameters of neural network randomly;
4: **for** $e = 1 : E$ **do**
5: **for** $b = 1 : M$ **do**
6: Forward propagate with the mini-batch of m_b samples in dataset;
7: Backward propagate network to get $\frac{\partial \mathcal{L}}{\partial w_1}$, $\frac{\partial \mathcal{L}}{\partial b_1}$, $\frac{\partial \mathcal{L}}{\partial w_2}$, $\frac{\partial \mathcal{L}}{\partial b_2}$ and gradients for weights and biases of previous layers;
8: Update w_1, b_1, w_2, b_2 and weights and biases of previous layers with obtained gradients;
9: **end**
10: **end**

IV. EXPERIMENTS

In this section, we make experiments on UCI datasets to validate the effectiveness of our proposed method and compare it with other algorithms including K nearest neighbors (KNN) algorithm, SVM [1], TSVM [8], NPHSVM [28], least squares recursive projection twin support vector machine (LSPTSVM) [29] and DSVM [25]. Our experiments are implemented on a Windows 10 computer with 3.6GHz Intel Core i7-9700K with 32GB RAM and RTX 2080Ti. Before training, as for KNN, SVM, TSVM, NPHSVM, LSPTSVM, values of data are located in [0,1] and values for model of DSVM and DNHSVM are standardized. The compared methods are briefly introduced as follows:

- KNN: k nearest neighbor algorithm, the core idea of the KNN algorithm is that if most of the k nearest samples in the feature space belong to a certain category, the sample also belongs to this category.
- SVM: it constructs a hyperplane with the structure risk minimization principle and parameters of hyperplane can be obtained by solving a QPP.
- TSVM: it aims to construct two non-parallel hyperplanes and each hyperplane is close to its own class and as far as possible to the other class, which can be solved by two small QPPs.
- NPHSVM: it constructs two non-parallel hyperplanes simultaneously by solving a single QPP and is consistent between its prediction and training processes.
- LSPTSVM: it adds a regularization term to projection twin support vector machine to ensure that the optimization problems are positive definite and owns better generalization ability.
- DSVM: it uses DNN to combine feature extraction with the task of constructing hyperplane seamlessly and the hinge loss is used to be the loss function.

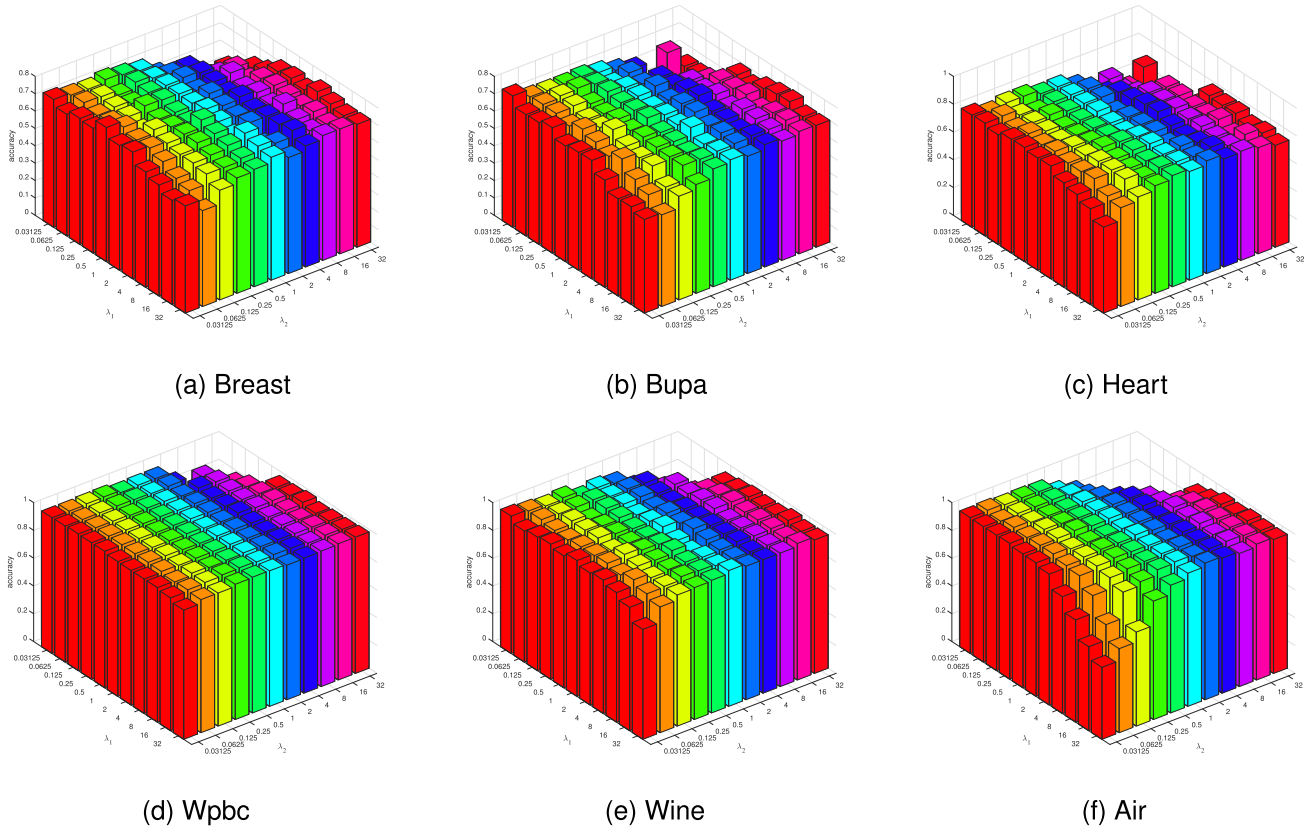

FIGURE 3. The accuracies versus λ_1 and λ_2 for UCI datasets.

TABLE 1. Characteristics of UCI datasets.

Datasets	classes	Examples	Attributes	Datasets	classes	Examples	Attributes
Austra	2	690	14	Vote	2	435	16
Breast	2	277	9	Wpbc	2	569	30
Bupa	2	345	9	Sonar	2	208	60
Cloud	2	1024	10	Wine	3	178	13
Ionosphere	2	351	32	Air	3	359	64
Heart	2	270	13	Vehicle	4	846	18

In order to show the performance and generalization of our and other compared algorithms, we perform experiments on UCI datasets and details of UCI datasets we use are shown in Table 1.

A. IMPLEMENTATION

In this subsection, we introduce the setting of all methods. For hyperparameters of all methods, they are determined by the strategy of cross-validation and chosen from the set $\{2^i | i = -5, -4, -3, \cdots, 3, 4, 5\}$. As for the nonlinear version of compared methods, RBF kernel is used to be the kernel function. SVM and TSVM we use in experiments are nonlinear versions. The number of neighbors for KNN algorithm is set to 3. Specifically, as for DSVM and our DNHSVM in experiments, we train neural network with one hidden layer with the number of neurons selected from the set {50, 100, 200, 500, 1000} and all layers are fully connected. The learning rate α is fixed to 0.01 and mini-batch m_b is set to 100. The maximum number of training epochs E in the optimization process is set to 100 and Adam optimizer is applied when training neural network and Relu is used to be the activation function. We repeat algorithms for five times and report average performance and standard deviation.

B. EXPERIMENT RESULTS

The results for all methods are shown in Table 2, and we report the best performance in boldface. As we can see from Table 2, our method obtains the optimal performance nearly all the datasets. To be specific, DNHSVM achieves

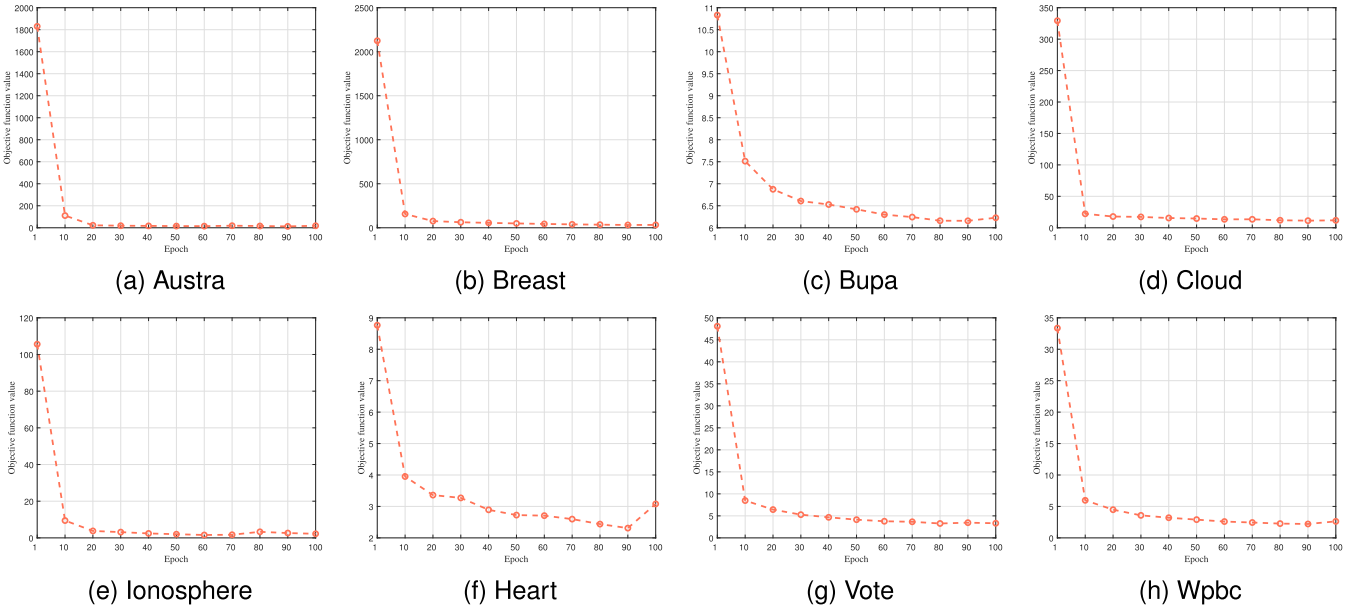

FIGURE 4. The objective function value for the increments of epochs on UCI datasets.

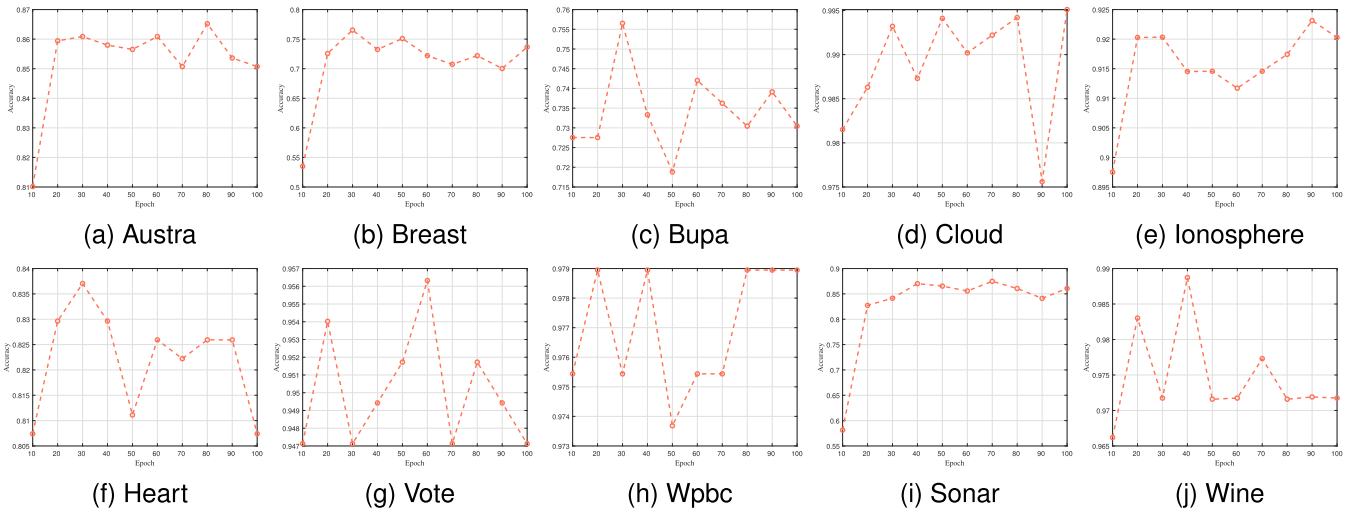

FIGURE 5. The accuracy for the increments of epochs on the UCI datasets.

about 0.58%, 1.09%, and 1.89% relative improvement over accuracy compared to the second best algorithm on datasets Austra, Breast, and Vehicle, respectively.

There are some reasons why our proposed method outperforms other state-of-the-art methods: (1) The deep model integrates feature extraction and construction of hyperplanes seamlessly, and samples can be easy to be classified. (2) The number of neurons is not fixed, and original data will be projected to a suitable space in which positive and negative samples have significant differences.

C. HYPERPARAMETER INFLUENCE AND CONVERGENCE ANALYSIS

The parameter influence analysis is shown in this part and parameter influence for UCI datasets Breast, Bupa, Heart, Wpbc, Wine and Air is shown in Fig. 3. As for other UCI datasets mentioned in Table 1, they are insensitive to the choice of hyperparameters. To this end, the influence of these datasets is not described in Fig. 3. As we can see from Fig. 3, our method achieves the promising performance when both λ_1 and λ_2 hold a small value. In addition, our method has a poor performance when λ_1 tends to a small value and λ_2 tends to a big value. Hence we suggest that setting parameter λ_1 and λ_2 to a small value is beneficial to the performance of our method.

The convergence is described in Fig. 4. We list convergence condition of binary classification datasets for our methods as the classification is to train multiple classifiers and their objective function value is just the sum of the multiple classifiers. As we can see from Fig. 4, the objective function value of our method is monotonically decreasing in the most datasets and tends to be stable after dozens of iterations.

TABLE 2. Comparison of classification performance (mean% ± standard deviation%) and training time on UCI datasets.

Datasets	KNN	SVM	TSVM	NPHSVM	LSPTSVM	DSVM	DNHSVM(ours)
Austra	67.83 ± 4.69	84.93±0.57	85.65±4.65	85.80±4.31	85.80±3.01	85.94±2.08	**86.52±2.36**
	-	0.00001	0.04207	0.3212	0.0056	0.3687	0.4467
Breast	72.86± 3.27	70.79±0.61	72.96±6.30	69.68±2.96	74.36±2.70	75.44±3.94	**76.53±5.35**
	-	0.000007	0.0095	0.0341	0.0039	0.2046	0.1811
Bupa	63.19±5.91	68.98±4.35	60.00±7.71	64.35±5.58	73.62±5.74	**75.65±6.70**	**75.65±6.82**
	-	0.000007	0.0130	0.0398	0.0034	0.2292	0.1888
Cloud	**99.51±0.44**	98.92±0.57	97.95±1.06	97.17±1.06	98.73±0.82	99.31±0.39	**99.51±0.44**
	-	0.000008	0.1081	0.3438	0.0060	0.4670	0.5789
Ionosphere	84.50±3.88	89.17±2.66	87.75±4.21	89.16±4.61	89.75±1.80	91.74±2.58	**92.31±2.11**
	-	0.000006	0.0143	0.0543	0.00610	0.2484	0.2955
Heart	80.37±1.89	81.85±4.40	82.59±4.26	81.48±4.14	83.33±2.93	81.85±4.89	**83.70±3.19**
	-	0.000006	0.0101	0.0393	0.0032	0.1898	0.1778
Vote	90.57±0.86	90.11±4.02	90.80±2.30	95.40±1.41	94.25±2.70	95.17±2.56	**95.63±1.98**
	-	0.000007	0.0234	0.0763	0.0048	0.2297	0.2909
Wpbc	92.80± 2.90	**97.89±1.19**	96.31±1.31	97.01±1.34	96.66±1.44	97.72±2.26	**97.89±2.64**
	-	0.000007	0.0326	0.1280	0.0047	0.3470	0.4280
Sonar	82.86±4.09	87.08±6.07	71.17±8.08	76.93±5.43	87.03±6.23	84.11±3.35	**87.48±2.90**
	-	0.000006	0.0089	0.0411	0.0044	0.1853	0.2249
Wine	73.33±5.72	93.79±3.33	98.30±2.55	**98.87±1.54**	98.86±1.56	97.73±2.13	**98.87±1.38**
	-	0.0022	0.0186	0.0247	0.0042	0.1783	0.5065
Air	83.61±4.68	94.98±1.93	98.05±1.25	96.65±3.49	95.27±2.10	98.04±1.42	**98.32±1.37**
	-	0.0058	0.0476	0.0370	0.1163	0.6059	0.8676
Vehicle	70.70±4.23	80.85±1.63	79.55±3.44	79.55±3.32	79.55±3.32	80.97±3.02	**82.86±1.51**
	-	0.0233	1.1848	0.06837	0.1712	1.3941	2.0923

TABLE 3. The *p*-value for the results of test datasets for ANOVA.

Ours V.S.	KNN	SVM	TSVM	NPHSVM	LSPTSVM	DSVM
p-value	0.0011	0.0008	0.0170	0.0082	0.0006	0.0059

D. DISCUSSION

In this subsection, we give a discussion of the proposed DNHSVM model. The results in Table 2 show that our method outperforms other state-of-the-art methods, which shows that the deep model can effectively extract friendly features and that better hyperplanes can be obtained for our method.

The influence of hyperparameter is described in Fig. 3. We discuss the influence of hyperparameters and convergence analysis in subsection IV-C and we give suggestions for the choice of hyperparameters. The figures of convergence show that the objective function value tends to be stable after dozens of iterations.

We report related results for the accuracy as the increments of epochs on UCI datasets in Fig. 5. As for datasets Austra, Breast, Ionosphere and Sonar, the performance of these datasets rises firstly and then fluctuates in a small range. For other datasets, the performance of our methods fluctuates on a relatively large scale.

Analysis of Variance (ANOVA) is a statistical hypothesis test, which can be used to analyze the accuracy results and test significance differences between several groups of results.

In order to evaluate whether the performance improvement observed from our proposed DNHSVM is statistically significant. The proposed DNHSVM and compared state-of-the-art methods are used to pairwise comparison on test datasets. The null hypothesis is that there is no significant difference in accuracy between the two methods on these datasets. The accuracy on test datasets are shown in Table 2, and we obtain the p-value shown in Table 3. Since all the p-values in Table 3 are less than 0.1, the significant difference is at the 0.1 significant level.

V. CONCLUSION

In this paper, we propose a novel deep non-parallel hyperplane support vector machine that combines feature extraction and classification seamlessly and the features which are extracted through neural networks are friendly for classification. As for the construction of hyperplanes, each hyperplane is closer to its own class and as far as possible to the other classes. The experiments performed on UCI datasets show that our proposed method outperforms other state-of-the-art methods, which shows the effectiveness of our proposed algorithms. In future work, we will extend our model to multi-view learning and semi-supervised learning.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[2] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.

[3] X. Huang, L. Shi, and J. A. K. Suykens, "Support vector machine classifier with pinball loss," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 984–997, May 2014.

[4] Y. Feng, Y. Yang, X. Huang, S. Mehrkanoon, and J. A. K. Suykens, "Robust support vector machines for classification with nonconvex and smooth losses," *Neural Comput.*, vol. 28, no. 6, pp. 1217–1247, 2016.

[5] O. Mangasarian and E. Wild, "Multisurface proximal support vector machine classification via generalized eigenvalues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 1, pp. 69–74, Jan. 2005.

[6] Y. Tian, Z. Qi, X. Ju, Y. Shi, and X. Liu, "Nonparallel support vector machines for pattern classification," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1067–1079, Jul. 2014.

[7] Y. H. Shao, C. N. Li, and N. Y. Deng, "Robust L1-norm non-parallel proximal support vector machine," *Optimization*, vol. 65, no. 1, pp. 169–183, 2016.

[8] Jayadeva, R. Khemchandani, and S. Chandra, "Twin support vector machines for pattern classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 905–910, Mar. 2007.

[9] S. Gao, Q. Ye, and N. Ye, "1-norm least squares twin support vector machines," *Neurocomputing*, vol. 74, no. 17, pp. 3590–3597, 2011.

[10] Z. Qi, Y. Tian, and Y. Shi, "Structural twin support vector machine for classification," *Knowl.-Based Syst.*, vol. 43, no. 2, pp. 74–81, May 2013.

[11] C. Wang, Q. Ye, P. Luo, N. Ye, and L. Fu, "Robust capped L1-norm twin support vector machine," *Neural Netw.*, vol. 114, pp. 47–59, Jun. 2019.

[12] X. Xie and S. Sun, "Multitask twin support vector machines," in *Proc. Int. Conf. Neural Inf. Process.*, 2012, pp. 341–348.

[13] X. Xie and S. Sun, "Multitask centroid twin support vector machines," *Neurocomputing*, vol. 149, pp. 1085–1091, Feb. 2015.

[14] X. Xie and S. Sun, "General multi-view semi-supervised least squares support vector machines with multi-manifold regularization," *Inf. Fusion*, vol. 62, pp. 63–72, Oct. 2020.

[15] X. Xie and S. Sun, "Multi-view twin support vector machines," *Intell. Data Anal.*, vol. 19, no. 4, pp. 701–712, 2015.

[16] X. Xie and S. Sun, "Multi-view support vector machines with the consensus and complementarity information," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 12, pp. 2401–2413, 2020.

[17] X. Xie, "Multi-view semi-supervised least squares twin support vector machines with manifold-preserving graph reduction," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 11, pp. 2489–2499, Nov. 2020.

[18] Y.-H. Shao, W.-J. Chen, J.-J. Zhang, Z. Wang, and N.-Y. Deng, "An efficient weighted Lagrangian twin support vector machine for imbalanced data classification," *Pattern Recognit.*, vol. 47, no. 9, pp. 3158–3167, Sep. 2014.

[19] C. Li, P. Ren, Y. Shao, Y. Ye, and Y. Guo, "Generalized elastic net Lp-norm nonparallel support vector machine," *Eng. Appl. Artif. Intell.*, vol. 88, Feb. 2020, Art. no. 103397.

[20] Y. Li, H. Sun, and W. Yan, "Domain adaptive twin support vector machine learning using privileged information," *Neurocomputing*, vol. 469, pp. 13–27, Jan. 2022.

[21] X. Xie, F. Sun, J. Qian, L. Guo, R. Zhang, X. Ye, and Z. Wang, "Laplacian Lp norm least squares twin support vector machine," *Pattern Recognit.*, vol. 136, Apr. 2023, Art. no. 109192.

[22] S. Rezvani and X. Wang, "Intuitionistic fuzzy twin support vector machines for imbalanced data," *Neurocomputing*, vol. 507, pp. 16–25, Oct. 2022.

[23] B. Chen, Y. Fan, W. Lan, J. Liu, C. Cao, and Y. Gao, "Fuzzy support vector machine with graph for classifying imbalanced datasets," *Neurocomputing*, vol. 514, pp. 296–312, Dec. 2022.

[24] D. Li, Y. Tian, and H. Xu, "Deep twin support vector machine," in *Proc. IEEE Int. Conf. Data Mining Workshop*, Dec. 2014, pp. 65–73.

[25] Y. Tang, "Deep learning using support vector machines," School Inf. Sci. Eng., College Comput. Eng., Ningbo Univ., Ningbo, China, Jimei Univ., Xiamen, China, Tech. Rep. 109192, 2013.

[26] C. Wu, S. Wang, B. Pan, and H. Chen, "Facial expression recognition with deep two-view support vector machine," in *Proc. 24th ACM Int. Conf. Multimedia*, 2016, pp. 616–620.

[27] O. Okwuashi and C. E. Ndehedehe, "Deep support vector machine for hyperspectral image classification," *Pattern Recognit.*, vol. 103, Jul. 2020, Art. no. 107298.

[28] Y.-H. Shao, W.-J. Chen, and N.-Y. Deng, "Nonparallel hyperplane support vector machine for binary classification problems," *Inf. Sci.*, vol. 263, pp. 22–35, Apr. 2014.

[29] Y. Shao, N. Deng, and Z. Yang, "Least squares recursive projection twin support vector machine for classification," *Pattern Recognit.*, vol. 45, no. 6, pp. 2299–2307, 2012.

FEIXIANG SUN received the B.E. degree from the Anhui University of Science and Technology, Huainan, China, in 2020. He is currently pursuing the master's degree with Ningbo University, Ningbo, China. His research interests include support vector machines, multi-view learning, and clustering.

XIJIONG XIE received the Ph.D. degree from the Pattern Recognition and Machine Learning Research Group, Department of Computer Science and Technology, East China Normal University, in 2016. He is currently an Associate Professor with the Faculty of Electrical Engineering and Computer Science, Ningbo University, China. He has over 30 publications at peer-reviewed journals and conferences in his research areas, such as IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), IEEE TRANSACTIONS ON CYBERNETICS, *Expert systems and application*, *Pattern Recognition*, *Neurocomputing*, and *Information Fusion*. His research interests include kernel methods, support vector machines, multi-view learning, and deep learning.