

A Nonlinear ν -Twin Support Tensor Machine

Hui-Ru Wang¹, Wei-Song Mu^{2,3} and Zhi-Jian Zhou^{1,*}

¹College of science; ²College of Information and Electrical Engineering
China Agricultural University,

Tsinghua East Road, Haidian District, Beijing, China

³Key Laboratory of Viticulture and Enology, Ministry of Agriculture
wanghuru529@163.com; wsmu@cau.edu.cn; zzjmath@163.com

Received December 2017; revised September 2018

ABSTRACT. *In image processing and face recognition, the data is stored in tensor form. In such case, the conventional vector-based approaches represented by ν -twin support vector machine (ν -TSVM) are not enough to meet the classification requirements. Therefore, many researchers concentrate on the study of tensor-based algorithms, while most of them are linear cases. In this paper, we propose a novel nonlinear ν -twin support tensor machine (NL ν -TSTM), which separates most samples by constructing two nonparallel hyperplanes in tensor kernel space. We use an alternating projection method to implement it. The new algorithm can retain more data structure information efficiently for it handles tensor data directly. Besides, it can overcome the overfitting problem which usually exists in vector-based approaches to some extent. The efficiency and superiority of the proposed method are demonstrated by computational experiments on different kinds of datasets.*

Keywords: Nonparallel hyperplane, Matrix kernel, Tensor learning, Classification

1. Introduction. Support vector machine (SVM) [1] has its unique properties for binary classification. To increase its solving speed, Jayadeva et al. [2] proposed twin support vector machine (TSVM) which seeks two nonparallel hyperplanes by solving two smaller-scale quadratic programming problems (QPPs). The solving speed of TSVM has been proved four times faster than SVM. In the past few decades, massive improvements based on TSVM have been proposed, such as ν -TSVM [3], least squares TSVH [4] and rough margin-based ν -TSVM [5, 6].

Tensor representation has been applied in numerous areas, especially in image processing for it can reduce the overfitting problem to a great extent. For instance, a second order tensor can express original gray image [7]. When dealing with tensor data, the aforementioned vector-based algorithms inevitably need to convert tensors into vectors before classification. This transformation can lead to structural information lose and data correlation damage [8]. What's more, the converted vectors are usually high dimensional which can easily lead to the overfitting problem and the curse of dimensionality problem.

To retain more structural information of tensors, a tensor-based algorithm, named linear support tensor machine (STM), has been put forward [9], which deals with input tensor data directly without vectorization. In addition, the experimental results verified that STM has a better performance compared with conventional SVM. In recent years, many researchers have been interested in extending the vector-based algorithms to tensor-based classification approaches, and many have gained good performance, such as linear ν -STM [8], proximal STM [10], higher rank STMs [11], higher order STM [12] and TSTM [13].

Moreover, several researchers also studied nonlinear conditions on kernel methods for tensors. Gao [14] proposed kernel support tensor regression where he applied kernel matrix [15] to deal with tensor data directly. He et al. [16] proposed a novel dual strategy in structure-preserving kernels and applied it to neuroimages. A nonlinear least squares TSTM was proposed for image classification [17]. Chen et al. [18] addressed one-class classification problem with the principle of maximal margin in tensor space.

Up to now, nonlinear classifiers based on tensor-kernel space are still rare and deserve further study. Accordingly, we propose a new tensor-kernel algorithm, named nonlinear ν -twin support tensor machine (NL ν -TSTM). Based on the bidirectional optimal projection algorithm, the optimal solutions of NL ν -TSTM can be obtained in an iterative manner. Similarly, the parameters ν still have their theoretical significance, i.e. they can control the bounds on the fractions of support tensors and error margins. The direct use of tensor-kernel representation reserves the structural information more efficiently. Compared with vector-based algorithms, the proposed NL ν -TSTM can avoid the overfitting problem to a large extent and is more suitable for the high-dimensional small sample size (HDS3) problem. The validity of the new algorithm is examined by numerous experiments.

The rest of paper is organized as follows. Section 2 outlines the ν -TSVM and tensor kernel matrix. Section 3 is our NL ν -TSTM algorithm. Section 4 shows the experimental results on vector-based datasets, and Section 5 considers tensor-based datasets. Finally, Section 6 concludes our works.

2. Related Work.

2.1. Nonlinear ν -Twin Support Vector Machine. The training dataset is $T = \{(\mathbf{x}_1, +1), \dots, (\mathbf{x}_p, +1), (\mathbf{x}_{p+1}, -1), \dots, (\mathbf{x}_{p+q}, -1)\}$, where $\mathbf{x}_i \in \mathbb{R}^n$. Matrix $\mathbf{A} \in \mathbb{R}^{p \times n}$, $\mathbf{B} \in \mathbb{R}^{q \times n}$ stands for the positive and negative data, respectively. Nonlinear ν -TSVM [3] considers two kernel-generated surfaces $K(\mathbf{x}^T, \mathbf{C}^T)\boldsymbol{\mu}_i + b_i = 0, i = \pm$, where $\mathbf{C}^T = [\mathbf{A}^T \ \mathbf{B}^T] \in \mathbb{R}^{n \times (p+q)}$ and K is an appropriately chosen kernel function. It constructs the following two QPPs,

$$\begin{aligned} \min_{\boldsymbol{\mu}_+, b_+, \rho_+, \boldsymbol{\xi}_-} \quad & \frac{1}{2} \left\| K(\mathbf{A}, \mathbf{C}^T)\boldsymbol{\mu}_+ + \mathbf{e}_+ b_+ \right\|^2 - \nu_1 \rho_+ + \frac{1}{q} \mathbf{e}_-^T \boldsymbol{\xi}_- \\ \text{s.t.} \quad & -(K(\mathbf{B}, \mathbf{C}^T)\boldsymbol{\mu}_+ + \mathbf{e}_- b_+) \geq \rho_+ \mathbf{e}_- - \boldsymbol{\xi}_-, \rho_+ \geq 0, \boldsymbol{\xi}_- \geq \mathbf{0}, \end{aligned} \quad (1)$$

and

$$\begin{aligned} \min_{\boldsymbol{\mu}_-, b_-, \rho_-, \boldsymbol{\xi}_+} \quad & \frac{1}{2} \left\| K(\mathbf{B}, \mathbf{C}^T)\boldsymbol{\mu}_- + \mathbf{e}_- b_- \right\|^2 - \nu_2 \rho_- + \frac{1}{p} \mathbf{e}_+^T \boldsymbol{\xi}_+ \\ \text{s.t.} \quad & K(\mathbf{A}, \mathbf{C}^T)\boldsymbol{\mu}_- + \mathbf{e}_+ b_- \geq \rho_- \mathbf{e}_+ - \boldsymbol{\xi}_+, \rho_- \geq 0, \boldsymbol{\xi}_+ \geq \mathbf{0}, \end{aligned} \quad (2)$$

where $\boldsymbol{\xi}_+$, $\boldsymbol{\xi}_-$ are slack vectors; \mathbf{e}_+ , \mathbf{e}_- are column vectors of ones. A new testing sample \mathbf{x} is assigned to class i ($i = +1, -1$) by

$$\text{class } i = \arg \min_{i=+,-} |K(\mathbf{x}^T, \mathbf{C}^T)\boldsymbol{\mu}_i + b_i| / \|\boldsymbol{\mu}_i\|. \quad (3)$$

2.2. Tensor Kernel Matrix. In vector-based classification problems, most datasets are nonlinear separable which can be solved by introducing a kernel function. Similarly, an important issue of nonlinear tensor-based classification problems is constructing appropriate kernel matrix. Up to now, the kernel matrix proposed by [14] has been extensively used for its simple calculation and preserving more structural information of tensors.

Let $\mathbf{X}_i \in \mathbb{R}^{n_1 \times n_2}$ represent a second order tensor and define $k : \mathbb{R}^{n_2} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}$ is a kernel function; $\mathcal{H} \in \mathbb{R}^\infty$ is the reproducing kernel Hilbert space of k and $\varphi : \mathbb{R}^{n_2} \rightarrow \mathbb{R}^\infty$ is the corresponding feature mapping. The nonlinear mapping function for matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$

is defined as: $\varphi(\mathbf{X}) = [\varphi(\mathbf{x}_i)] \in \mathbb{R}^{n_1 \times \infty}$, where $\mathbf{x}_i \in \mathbb{R}^{n_2}$ is the i th row of \mathbf{X} . Therefore, for arbitrary matrix \mathbf{X} , $\mathbf{Z} \in \mathbb{R}^{n_1 \times n_2}$, the tensor kernel matrix is denoted as:

$$K_{\mathbf{XZ}} = \varphi(\mathbf{X})\varphi(\mathbf{Z})^T = \left[\varphi(\mathbf{x}_i)\varphi(\mathbf{z}_j)^T \right]_{n_1 \times n_1}, \quad (4)$$

where $\varphi(\mathbf{x}_i)\varphi(\mathbf{z}_j)^T = e^{-\|\mathbf{x}_i - \mathbf{z}_j\|^2/2r^2}$ can be a Gaussian radial basis function (RBF). It is clear that the value of kernel function in vector space is a scalar while it is a matrix in tensor space. We call $K_{\mathbf{XZ}}$ as the TRBF kernel matrix and define $k(\mathbf{x}_i, \mathbf{z}_j) = \varphi(\mathbf{x}_i)\varphi(\mathbf{z}_j)^T$ for simplification.

3. Nonlinear ν -Twin Support Tensor Machine. As mentioned above, the vectorization neglects the latent structural information of tensor data. The converted vector is usually high-dimensional which can easily leads to curse of dimensionality. Till now, there are still few researches on nonlinear classifier in tensor learning. Therefore, we propose our NL ν -TSTM in the following subsections.

3.1. NL ν -TSTM and Its Algorithm. Suppose we are given the training dataset $T = \{(\mathbf{X}_1, +1), \dots, (\mathbf{X}_p, +1), (\mathbf{X}_{p+1}, -1), \dots, (\mathbf{X}_{p+q}, -1)\}$. The essential principal of NL ν -TSTM is to seek the two following nonparallel hyperplanes:

$$\mathbf{u}^T \varphi(\mathbf{X}) \mathbf{v} + b_+ = 0 \quad \text{and} \quad \tilde{\mathbf{u}}^T \varphi(\mathbf{X}) \tilde{\mathbf{v}} + b_- = 0, \quad (5)$$

where $\mathbf{u}, \tilde{\mathbf{u}} \in \mathbb{R}^{n_1}$, $\mathbf{v}, \tilde{\mathbf{v}} \in \mathbb{R}^\infty$ and $b_+, b_- \in \mathbb{R}$. Let $f_+(\mathbf{X}) = \mathbf{u}^T \varphi(\mathbf{X}) \mathbf{v} + b_+$, $f_-(\mathbf{X}) = \tilde{\mathbf{u}}^T \varphi(\mathbf{X}) \tilde{\mathbf{v}} + b_-$. Then, a new tensor sample \mathbf{X} is assigned by:

$$\text{class } i = \arg \min_{i=+,-} |f_i(\mathbf{X})| / \|\mathbf{u}_i \mathbf{v}_i^T\|, \quad (6)$$

where $\mathbf{u} = \mathbf{u}_+$, $\mathbf{v} = \mathbf{v}_+$, $\tilde{\mathbf{u}} = \mathbf{u}_-$, $\tilde{\mathbf{v}} = \mathbf{v}_-$.

The two QPPs of NL ν -TSTM are denoted as follows:

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{v}, b_+, \rho_+, \xi_j} & \frac{1}{2} \sum_{i=1}^p (\mathbf{u}^T \varphi(\mathbf{X}_i) \mathbf{v} + b_+)^2 - \nu_1 \rho_+ + \frac{1}{q} \sum_{j=p+1}^{p+q} \xi_j \\ \text{s.t.} & -(\mathbf{u}^T \varphi(\mathbf{X}_j) \mathbf{v} + b_+) \geq \rho_+ - \xi_j, \rho_+ \geq 0, \xi_j \geq 0, j = p+1, \dots, p+q, \end{aligned} \quad (7)$$

and

$$\begin{aligned} \min_{\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, b_-, \rho_-, \xi_i} & \frac{1}{2} \sum_{j=p+1}^{p+q} (\tilde{\mathbf{u}}^T \varphi(\mathbf{X}_j) \tilde{\mathbf{v}} + b_-)^2 - \nu_2 \rho_- + \frac{1}{p} \sum_{i=1}^p \xi_i \\ \text{s.t.} & (\tilde{\mathbf{u}}^T \varphi(\mathbf{X}_i) \tilde{\mathbf{v}} + b_-) \geq \rho_- - \xi_i, \rho_- \geq 0, \xi_i \geq 0, i = 1, \dots, p, \end{aligned} \quad (8)$$

where ξ_i, ξ_j are slack variables; ν_1, ν_2 are new parameters; ρ_+, ρ_- are additional variables. It is note that for all $\xi_j = 0, j = p+1, \dots, p+q$ (or $\xi_i = 0, i = 1, \dots, p$), the negative (or positive) samples are separated by the positive (or negative) hyperplane, with the margin $\rho_+ / \|\mathbf{u} \mathbf{v}^T\|^2$ (or $\rho_- / \|\tilde{\mathbf{u}} \tilde{\mathbf{v}}^T\|^2$). By introducing the Lagrangian multipliers η_j, r_j, s into QPP (7), we can drive its Lagrangian function:

$$\begin{aligned} \mathcal{L}(\mathbf{u}, \mathbf{v}, b_+, \rho_+, \xi_j, \eta_j, r_j, s) &= \frac{1}{2} \sum_{i=1}^p (\mathbf{u}^T \varphi(\mathbf{X}_i) \mathbf{v} + b_+)^2 - \nu_1 \rho_+ + \frac{1}{q} \sum_{j=p+1}^{p+q} \xi_j - s \rho_+ \\ &\quad - \sum_{j=p+1}^{p+q} \eta_j [-(\mathbf{u}^T \varphi(\mathbf{X}_j) \mathbf{v} + b_+) - \rho_+ + \xi_j] - \sum_{j=p+1}^{p+q} r_j \xi_j. \end{aligned} \quad (9)$$

Differentiating the Lagrangian function \mathcal{L} with respect to variables $\mathbf{u}, \mathbf{v}, b_+, \rho_+, \xi_j$ can obtain the following equations,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \sum_{i,k=1}^p \varphi(\mathbf{X}_k) \mathbf{v} (\mathbf{u}^T \varphi(\mathbf{X}_i) \mathbf{v} + b_+)^T + \sum_{j=p+1}^{p+q} \eta_j (\varphi(\mathbf{X}_j) \mathbf{v}) = 0, \quad (10)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \sum_{i,k=1}^p (\mathbf{u}^T \varphi(\mathbf{X}_k))^T (\mathbf{u}^T \varphi(\mathbf{X}_k) \mathbf{v} + b_+) + \sum_{j=p+1}^{p+q} \eta_j (\mathbf{u}^T \varphi(\mathbf{X}_j))^T = 0, \quad (11)$$

$$\frac{\partial \mathcal{L}}{\partial b_+} = \sum_{i=1}^p (\mathbf{u}^T \varphi(\mathbf{X}_i) \mathbf{v} + b_+) + \sum_{j=p+1}^{p+q} \eta_j = 0, \quad (12)$$

$$\frac{\partial \mathcal{L}}{\partial \rho_+} = \eta_j - \nu_1 - s = 0, \quad (13)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_j} = 1/q - \eta_j - r_j = 0. \quad (14)$$

From Eqs. (10) and (11), we know that \mathbf{u}, \mathbf{v} are dependent on each other and cannot be solved independently. Therefore, we adopt the alternating projection method to solve it.

For any given column vector $\mathbf{u} \in \mathbb{R}^{n_1}$, we firstly define a new matrix kernel function $k_{\mathbf{u}} : \mathbb{R}^{n_1 \times n_2} \otimes \mathbb{R}^{n_1 \times n_2} \rightarrow \mathbb{R}$ with respect to \mathbf{u} as: $k_{\mathbf{u}}(\mathbf{X}, \mathbf{Z}) = \langle \mathbf{u}^T \varphi(\mathbf{X}), \mathbf{u}^T \varphi(\mathbf{Z}) \rangle = \mathbf{u}^T \varphi(\mathbf{X}) \varphi(\mathbf{Z})^T \mathbf{u}$. If we set $\mathcal{X} = \{\mathbf{X}_1, \dots, \mathbf{X}_s\}$ and $\mathcal{Z} = \{\mathbf{Z}_1, \dots, \mathbf{Z}_m\}$, where \mathcal{X}, \mathcal{Z} are third order tensor and define

$$K_{\mathbf{u}}(\mathcal{X}, \mathcal{Z}) = \begin{bmatrix} k_{\mathbf{u}}(\mathbf{X}_1, \mathbf{Z}_1) & \cdots & k_{\mathbf{u}}(\mathbf{X}_1, \mathbf{Z}_m) \\ & \ddots & \\ k_{\mathbf{u}}(\mathbf{X}_s, \mathbf{Z}_1) & \cdots & k_{\mathbf{u}}(\mathbf{X}_s, \mathbf{Z}_m) \end{bmatrix} \in \mathbb{R}^{s \times m}.$$

Then for $\mathcal{A} = \{\mathbf{X}_1, \dots, \mathbf{X}_p\}$, $\mathcal{B} = \{\mathbf{X}_{p+1}, \dots, \mathbf{X}_{p+q}\}$, and $\mathcal{C} = \{\mathbf{X}_1, \dots, \mathbf{X}_{p+q}\}$, we get $K_{\mathbf{u}}(\mathcal{A}, \mathcal{C}) \in \mathbb{R}^{p \times (p+q)}$, $K_{\mathbf{u}}(\mathcal{B}, \mathcal{C}) \in \mathbb{R}^{q \times (p+q)}$ and $K_{\mathbf{u}}(\mathcal{C}, \mathcal{C}) \in \mathbb{R}^{(p+q) \times (p+q)}$.

For the sake of solving QPP (7), we firstly fix $\mathbf{u} \in \mathbb{R}^{n_1}$ and define:

$$\varphi_{\mathbf{u}}(\mathcal{X}) = \begin{bmatrix} (\varphi(\mathbf{X}_1)^T \mathbf{u})^T \\ \vdots \\ (\varphi(\mathbf{X}_s)^T \mathbf{u})^T \end{bmatrix} \in \mathbb{R}^{s \times \infty},$$

then $\varphi_{\mathbf{u}}(\mathcal{A}) \in \mathbb{R}^{p \times \infty}$, $\varphi_{\mathbf{u}}(\mathcal{B}) \in \mathbb{R}^{q \times \infty}$ and $\varphi_{\mathbf{u}}(\mathcal{C}) \in \mathbb{R}^{(p+q) \times \infty}$. Then, QPP (7) becomes

$$\begin{aligned} \min_{\mathbf{v}, b_+, \rho_+, \xi_-} & \frac{1}{2} \|\varphi_{\mathbf{u}}(\mathcal{A}) \mathbf{v} + \mathbf{e}_+ b_+\| - \nu_1 \rho_+ + \frac{1}{q} \mathbf{e}_-^T \xi_- \\ \text{s.t.} & -(\varphi_{\mathbf{u}}(\mathcal{B}) \mathbf{v} + \mathbf{e}_- b_+) \geq \rho_+ \mathbf{e}_- - \xi_-, \rho_+ \geq 0, \xi_- \geq \mathbf{0}, \end{aligned} \quad (15)$$

where $\xi_- = (\xi_{p+1}, \dots, \xi_{p+q})^T$. We see \mathbf{v} in the subspace $\text{span} \{\mathbf{u}^T \varphi(\mathbf{X}_1), \dots, \mathbf{u}^T \varphi(\mathbf{X}_{p+q})\}^T$ of \mathbb{R}^{∞} and assume $\mathbf{v} = \varphi_{\mathbf{u}}(\mathcal{C})^T \beta_+ \in \mathbb{R}^{\infty}$, where $\beta_+ \in \mathbb{R}^{p+q}$, then

$$\varphi_{\mathbf{u}}(\mathcal{A}) \mathbf{v} = K_{\mathbf{u}}(\mathcal{A}, \mathcal{C}) \beta_+ \in \mathbb{R}^{p \times 1}, \varphi_{\mathbf{u}}(\mathcal{B}) \mathbf{v} = K_{\mathbf{u}}(\mathcal{B}, \mathcal{C}) \beta_+ \in \mathbb{R}^{q \times 1}.$$

QPP (15) can be rewritten as:

$$\begin{aligned} \min_{\beta_+, b_+, \rho_+, \xi_-} & \frac{1}{2} \|K_{\mathbf{u}}(\mathcal{A}, \mathcal{C}) \beta_+ + \mathbf{e}_+ b_+\| - \nu_1 \rho_+ + \frac{1}{q} \mathbf{e}_-^T \xi_- \\ \text{s.t.} & -(K_{\mathbf{u}}(\mathcal{B}, \mathcal{C}) \beta_+ + \mathbf{e}_- b_+) \geq \rho_+ \mathbf{e}_- - \xi_-, \rho_+ \geq 0, \xi_- \geq \mathbf{0}. \end{aligned} \quad (16)$$

Obviously, QPP (16) is a traditional NL ν -TSVM problem, and its dual problem is

$$\begin{aligned} \min_{\boldsymbol{\eta}_+} \quad & \frac{1}{2} \boldsymbol{\eta}_+^T \mathbf{G}_u (\mathbf{H}_u^T \mathbf{H}_u)^{-1} \mathbf{G}_u^T \boldsymbol{\eta}_+ \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\eta}_+ \leq 1/q \cdot \mathbf{e}_-, \quad \mathbf{e}_-^T \boldsymbol{\eta}_+ \geq \nu_1, \end{aligned} \quad (17)$$

where $\mathbf{H}_u = [K_u(\mathcal{A}, \mathcal{C}) \quad \mathbf{e}_+]$, $\mathbf{G}_u = [K_u(\mathcal{B}, \mathcal{C}) \quad \mathbf{e}_-]$. The optimal solution is:

$$[\boldsymbol{\beta}_+^T, b_+]^T = -(\mathbf{H}_u^T \mathbf{H}_u)^{-1} \mathbf{G}_u^T \boldsymbol{\eta}_+. \quad (18)$$

Secondly, we calculate \mathbf{u} . For any given $\mathbf{v} \in \mathbb{R}^\infty$, there exists $\boldsymbol{\beta}_+ \in \mathbb{R}^{p+q}$ such that $\mathbf{v} = \varphi_u(\mathcal{C})^T \boldsymbol{\beta}_+$, and we define

$$\varphi_v(\mathcal{X}) = \begin{bmatrix} (\varphi(\mathbf{X}_1) \mathbf{v})^T \\ \vdots \\ (\varphi(\mathbf{X}_s) \mathbf{v})^T \end{bmatrix} \in \mathbb{R}^{s \times n_1}$$

then $\varphi_v(\mathcal{A}) \in \mathbb{R}^{p \times n_1}$, $\varphi_v(\mathcal{B}) \in \mathbb{R}^{q \times n_1}$ and QPP (7) is converted as:

$$\begin{aligned} \min_{\mathbf{u}, b_+, \rho_+, \boldsymbol{\xi}_-} \quad & \frac{1}{2} \|\varphi_v(\mathcal{A}) \mathbf{u} + b_+\| - \nu_1 \rho_+ + \frac{1}{q} \mathbf{e}_-^T \boldsymbol{\xi}_- \\ \text{s.t.} \quad & -(\varphi_v(\mathcal{B}) \mathbf{v} + \mathbf{e}_- b_+) \geq \rho_+ \mathbf{e}_- - \boldsymbol{\xi}_-, \quad \rho_+ \geq 0, \quad \boldsymbol{\xi}_- \geq \mathbf{0}. \end{aligned} \quad (19)$$

Similarly, we derive its dual problem.

$$\begin{aligned} \min_{\boldsymbol{\alpha}_+} \quad & \frac{1}{2} \boldsymbol{\alpha}_+^T \mathbf{G}_{\beta_+} (\mathbf{H}_{\beta_+}^T \mathbf{H}_{\beta_+})^{-1} \mathbf{G}_{\beta_+}^T \boldsymbol{\alpha}_+ \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha}_+ \leq 1/q \cdot \mathbf{e}_-, \quad \mathbf{e}_-^T \boldsymbol{\alpha}_+ \geq \nu_1, \end{aligned} \quad (20)$$

where $\mathbf{H}_{\beta_+} = [\varphi_{\beta_+}(\mathcal{A}) \quad \mathbf{e}_+]$ and $\mathbf{G}_{\beta_+} = [\varphi_{\beta_+}(\mathcal{B}) \quad \mathbf{e}_-]$,

$$\varphi_{\beta_+}(\mathcal{A}) = \begin{bmatrix} (\varphi(\mathbf{X}_1) \varphi_u(\mathcal{C})^T \boldsymbol{\beta}_+)^T \\ \vdots \\ (\varphi(\mathbf{X}_p) \varphi_u(\mathcal{C})^T \boldsymbol{\beta}_+)^T \end{bmatrix}, \quad \varphi_{\beta_+}(\mathcal{B}) = \begin{bmatrix} (\varphi(\mathbf{X}_{p+1}) \varphi_u(\mathcal{C})^T \boldsymbol{\beta}_+)^T \\ \vdots \\ (\varphi(\mathbf{X}_{p+q}) \varphi_u(\mathcal{C})^T \boldsymbol{\beta}_+)^T \end{bmatrix},$$

$$\varphi(\mathbf{X}_k) \varphi_u(\mathcal{C})^T = \varphi(\mathbf{X}_k) [\varphi(\mathbf{X}_1)^T \mathbf{u}, \dots, \varphi(\mathbf{X}_{p+q})^T \mathbf{u}] = [K_{\mathbf{X}_k \mathbf{X}_1} \mathbf{u}, \dots, K_{\mathbf{X}_k \mathbf{X}_{p+q}} \mathbf{u}] \in \mathbb{R}^{n_1 \times (p+q)}.$$

Then, the optimal (\mathbf{u}_+, b_+) can be obtained,

$$[\mathbf{u}^T, b_+]^T = -(\mathbf{H}_{\beta_+}^T \mathbf{H}_{\beta_+})^{-1} \mathbf{G}_{\beta_+}^T \boldsymbol{\alpha}_+. \quad (21)$$

From $\|\mathbf{v}\|^2 = \boldsymbol{\beta}_+^T \varphi_u(\mathcal{C}) \varphi_u(\mathcal{C})^T \boldsymbol{\beta}_+ = \boldsymbol{\beta}_+^T K_u(\mathcal{C}, \mathcal{C}) \boldsymbol{\beta}_+$ and $\|\mathbf{u} \mathbf{v}^T\|^2 = \|\mathbf{u}\|^2 \|\mathbf{v}\|^2$, we can get the value of $\|\mathbf{u} \mathbf{v}^T\|^2$, and

$$f_+(\mathbf{X}) = [\mathbf{u}^T K_{\mathbf{X} \mathbf{X}_1} \mathbf{u}, \dots, \mathbf{u}^T K_{\mathbf{X} \mathbf{X}_{p+q}} \mathbf{u}] \boldsymbol{\beta}_+ + b_+, \quad (22)$$

$$\rho_+ = -\frac{1}{q_1} \sum_{j=1}^{q_1} ([\mathbf{u}^T K_{\mathbf{X}_j \mathbf{X}_1} \mathbf{u}, \dots, \mathbf{u}^T K_{\mathbf{X}_j \mathbf{X}_{p+q}} \mathbf{u}] \boldsymbol{\beta}_+ + b_+), \quad (23)$$

where q_1 denotes the number of $\mathbf{X}_j, j \in \{p+1, \dots, p+q\}$ satisfied with $0 < \alpha_j < 1/q$.

With similar steps, we can derive the value $(\boldsymbol{\beta}_-, \tilde{\mathbf{u}}, b_-)$, then ρ_- is calculated by

$$\rho_- = \frac{1}{p_1} \sum_{i=1}^{p_1} ([\tilde{\mathbf{u}}^T K_{\mathbf{X}_i \mathbf{X}_1} \tilde{\mathbf{u}}, \dots, \tilde{\mathbf{u}}^T K_{\mathbf{X}_i \mathbf{X}_{p+q}} \tilde{\mathbf{u}}] \boldsymbol{\beta}_- + b_-), \quad (24)$$

where p_1 denotes the number of $\mathbf{X}_i, i \in \{1, \dots, p\}$ satisfied with $0 < \alpha_i < 1/p$, and

$$f_-(\mathbf{X}) = [\tilde{\mathbf{u}}^T K_{\mathbf{X} \mathbf{X}_1} \tilde{\mathbf{u}}, \dots, \tilde{\mathbf{u}}^T K_{\mathbf{X} \mathbf{X}_{p+q}} \tilde{\mathbf{u}}] \boldsymbol{\beta}_- + b_-. \quad (25)$$

The flowchart of NL ν -TSTM is described specific as follows.

Algorithm 1. NL ν -TSTM

Inputs: the value ν_1, ν_2 , the maximum number of iteration I_+, I_- , the training samples $\mathbf{X}_i \in \mathbb{R}^{n_1 \times n_2} (i = 1, \dots, p + q)$ and testing samples $\mathbf{X}_j \in \mathbb{R}^{n_1 \times n_2} (j = 1, \dots, m)$

Outputs: the optimal $(\mathbf{u}, \boldsymbol{\beta}_+, b_+, \rho_+)$ and $(\tilde{\mathbf{u}}, \boldsymbol{\beta}_-, b_-, \rho_-)$, the labels of testing samples.

Step1: Initialization. Let $\mathbf{u}^t = (1, \dots, 1)^T$, $\tilde{\mathbf{u}}^t = (1, \dots, 1)^T$ and $\varepsilon > 0$ is small enough.

Step2: Calculate $(\boldsymbol{\beta}_+, b_+)$. Solving QPP (17) with $\mathbf{u} = \mathbf{u}^t$, and get $\boldsymbol{\eta}_+^t$, then $(\boldsymbol{\beta}_+^t, b_+^t)$ can be obtained by solving Eq.(18) with $\boldsymbol{\eta}_+ = \boldsymbol{\eta}_+^t$.

Step3: Update (\mathbf{u}, b_+) . After acquiring $\boldsymbol{\beta}_+ = \boldsymbol{\beta}_+^t$ in step2 and $\boldsymbol{\alpha}_+^t$ can be obtained by solving QPP (20), then solving Eq. (21) with $\boldsymbol{\alpha}_+ = \boldsymbol{\alpha}_+^t$ can get (\mathbf{u}, b_+) .

Step4: Compute \mathbf{u} and $\boldsymbol{\beta}_+$ iteratively from Step 2 \sim 3. If the following conditions: $\|\mathbf{u}^t - \mathbf{u}^{t-1}\| \leq \varepsilon$, $\|\boldsymbol{\beta}_+^t - \boldsymbol{\beta}_+^{t-1}\| \leq \varepsilon$ and $\|b_+^t - b_+^{t-1}\| \leq \varepsilon$ are satisfied simultaneously, or the iteration number exceeds the maximum number I_+ , the iteration will be terminated. After acquiring the optimal $(\mathbf{u}^*, \boldsymbol{\beta}_+^*, b_+^*)$, we get ρ_+ by solving Eq.(23).

Step5: Do the similar steps 2 \sim 4, $(\tilde{\mathbf{u}}^*, \boldsymbol{\beta}_-^*, b_-^*, \rho_-)$ can be acquired.

Step6: Calculate $\|\mathbf{u}\mathbf{v}^T\|^2$ and $\|\tilde{\mathbf{u}}\tilde{\mathbf{v}}^T\|^2$.

Step7: For a new sample, calculate $f_+(\mathbf{X})$ by Eq.(22) and $f_-(\mathbf{X})$ by Eq.(25).

Step8: Output the label of the new sample by (6).

3.2. Theoretical Interpretation. For simplicity, we still use the positive hyperplane to interpret. After solving (20) in the last iteration step, the optimal $\boldsymbol{\alpha}^* = (\alpha_{p+1}^*, \alpha_{p+2}^*, \dots, \alpha_{p+q}^*)^T$ can be acquired, then we can get the following propositions.

Proposition 3.1. *The negative samples can be divided into three conditions according to the corresponding values of α_j^* .*

- 1). If $\alpha_j^* = 0$, then the corresponding negative samples satisfy $\mathbf{u}^T \varphi(\mathbf{X}) \varphi_{\mathbf{u}}(\mathcal{C})^T \boldsymbol{\beta}_+ + b_+ < -\rho_+$. They are the negative samples which are classified absolutely right.
- 2). If $0 < \alpha_j^* < 1/q$ which implies the corresponding $\xi_j = 0$, these negative samples are support tensors on the hyperplane $\mathbf{u}^T \varphi(\mathbf{X}) \varphi_{\mathbf{u}}(\mathcal{C})^T \boldsymbol{\beta}_+ + b_+ = -\rho_+$.
- 3). If $\alpha_j^* = 1/q$, then the corresponding negative samples satisfy $\mathbf{u}^T \varphi(\mathbf{X}) \varphi_{\mathbf{u}}(\mathcal{C})^T \boldsymbol{\beta}_+ + b_+ > -\rho_+$. They are usually the outliers or noises of negative class.

Proposition 3.2. *Suppose we run QPP (7) with $(p + q)$ samples, acquiring ρ_+ , then,*

- 1). ν_1 is an upper bound on the fraction of negative margin errors.
- 2). ν_1 is a lower bound on the fraction of negative support tensors.

3.3. Convergence Analysis.

Theorem 3.1. *Using Algorithm 1, one can find the optimal $(\mathbf{u}^*, \boldsymbol{\beta}_+^*, b_+^*)$ and $(\tilde{\mathbf{u}}^*, \boldsymbol{\beta}_-^*, b_-^*)$, then Algorithm 1 is convergent.*

Proof: Let $f_1(\mathbf{u}, \mathbf{v}, b_+)$ be the objective function of QPP (7) and $\mathbf{v} = \varphi_{\mathbf{u}}(\mathcal{C})^T \boldsymbol{\beta}_+$, then the objective function f_1 can be rewrite as:

$$f_1(\mathbf{u}, \boldsymbol{\beta}_+, b_+) = \frac{1}{2} \sum_{i=1}^p (\mathbf{u}^T \varphi(\mathbf{X}) \varphi_{\mathbf{u}}(\mathcal{C})^T \boldsymbol{\beta}_+ + b_+)^2 - \nu_1 \rho_+ + \frac{1}{q} \sum_{j=p+1}^{p+q} \xi_j$$

Similar as [13], we can obtain a monotone decreasing sequences with lower bounds, and then prove Algorithm 1 is convergent. \square

4. Experimental Results on Vector-based Datasets. As we know, a vector can be regarded as a first order tensor. In this section, we do experiments to verify that the NL ν -TSTM has the ability to handle vector-based datasets. We firstly use Australian

dataset as an example to do the experiment, then we make a comprehensive comparison on six vector-based datasets ¹.

4.1. Datasets Description and Experiment Setting. In all experiments, we concentrate on the study of small-sized training sets to confirm the effectiveness of four algorithms. We apply grid search to select the optimal parameters. For a same parameter, the experiments are repeated 5 times, where one experiment is conducted as follows: we randomly choose 20% samples for training and the rest for testing. We set $\nu_1 = \nu_2 = \nu$ and ν is searched in $\{0.1, 0.2, \dots, 0.9\}$. The optimal RBF and TRBF kernel parameters are searched in $r = [2^{-10}, 2^{-9}, \dots, 2^{10}]$. All algorithms are carried out in Matlab 2014a operated on Windows 7 personal computer with 4.0 GB of RAM.

4.2. Experiments on Australian Dataset. Australian dataset has 690 samples with 14 attributes, and its features are scaled to $[-1, 1]$. As far as we know, a vector $\mathbf{x} \in \mathbb{R}^n$ can be convert to different kinds of matrix form $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$, where $n_1 \times n_2 \approx n$. There will be many combinations of n_1 and n_2 that satisfy the conditions. As suggested by [9], there are five possible tensor sizes in Australian dataset. It is significant to find which conversion is the best one.

The results with respect to different tensor sizes are shown in Table 1 and the bold one is the best, where ‘Num’ represents the size of training samples. The results of 1×14 indicate tensor-based algorithm can handle vector-based datasets directly, while the testing accuracies are a little bit lower compared with the results under other five types of matrix sizes. When tensor size is 4×4 , both $L\nu$ -TSTM and $NL\nu$ -TSTM obtain pleasurable performance. The experimental results indicate that the closer of n_1 and n_2 is, the better classification performance for tensor-based algorithms obtain. In addition, we find that the testing accuracies of $L\nu$ -TSTM and $NL\nu$ -TSTM improve as the the number of training samples increase under the same tensor size.

TABLE 1. Averaged testing accuracy in different tensor sizes using ν -TSTM.

Num	Algorithm	1×14	2×7	3×5	4×4	5×3	7×2
10	$L\nu$ -TSTM	-	84.36±2.33	86.98±3.92	86.41±2.73	83.94±2.71	84.78±0.90
	$NL\nu$ -TSTM	85.64±1.92	88.45±2.95	89.16±1.50	93.37±0.38	91.22±1.63	85.58±3.18
20	$L\nu$ -TSTM	86.49±4.64	85.57±1.36	84.98±0.88	85.35±0.39	85.23±0.54	85.08±0.47
	$NL\nu$ -TSTM	88.80±1.29	91.26±1.47	90.83±1.18	94.15±0.21	92.67±0.62	86.62±1.61
30	$L\nu$ -TSTM	86.25±1.80	86.32±2.71	85.26±0.54	86.51±2.33	85.71±1.22	85.11±0.76
	$NL\nu$ -TSTM	89.75±1.40	93.07±0.85	92.53±1.09	94.69±0.81	93.84±0.34	90.86±1.19
40	$L\nu$ -TSTM	86.95±1.87	86.09±1.45	86.09±0.86	87.01±0.82	85.67±0.82	85.51±0.47
	$NL\nu$ -TSTM	91.77±1.39	94.29±0.78	93.86±1.09	94.72±0.37	94.49±0.96	91.08±0.64

TABLE 2. Averaged testing accuracy and Gmeans on Australian dataset.

Num	$L\nu$ -TSVM		$L\nu$ -TSTM		$NL\nu$ -TSVM		$NL\nu$ -TSTM	
	Accuracy	Gmeans	Accuracy	Gmeans	Accuracy	Gmeans	Accuracy	Gmeans
10	82.51±4.93	83.76	86.41±2.73	87.25	85.67±3.08	85.91	93.37±0.38	93.31
20	84.21±1.60	84.32	85.35±0.39	85.41	88.31±2.16	88.40	94.15±0.21	94.18
30	86.38±1.07	86.41	86.51±2.33	86.43	90.31±1.88	90.26	94.69±0.81	94.56
40	86.52±1.36	86.44	87.01±0.82	87.03	90.06±0.57	89.94	94.72±0.37	94.60

The classification results of $NL\nu$ -TSTM, $NL\nu$ -TSVM, $L\nu$ -TSVM and $L\nu$ -TSTM with different training sample sizes are shown in Table 2. The results indicate that our $NL\nu$ -TSTM is outstanding from the point of testing accuracy. As the training number increases, the testing accuracies mostly arise, and $NL\nu$ -TSTM performs the best, followed

¹<http://www.cs.nyu.edu/~roweis/data.html>










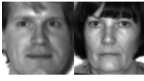

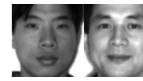

by $NL\nu$ -TSVM, $L\nu$ -TSTM and $L\nu$ -TSVM. When the number of training samples is 10, the accuracy of our $NL\nu$ -TSTM is 93.37%, which is much higher than $NL\nu$ -TSVM. Besides, when the tensor size is 1×14 and the size of training samples is 40, the accuracy of $NL\nu$ -TSTM is much higher than the other three algorithms. In general, when vector is regarded as first order tensor, our $NL\nu$ -TSTM has a comparable performance with $NL\nu$ -TSVM.

Gmeans is a commonly used index to evaluate the performance of algorithms on preventing the overfitting problem, especially in HDS3 problem [19]. Accordingly, the results of Gmeans show that $NL\nu$ -TSTM performs the best, which indicates that tensor-based algorithm ν -TSTM has promising superiorities on avoiding the overfitting problem.

TABLE 3. The results of six vector-based datasets.

Datasets	$L\nu$ -TSVM		$L\nu$ -TSTM		$NL\nu$ -TSVM		$NL\nu$ -TSTM	
	Accuracy	Gmeans	Accuracy	Gmeans	Accuracy	Gmeans	Accuracy	Gmeans
	ν	Time(s)	ν	Time(s)	(ν, r)	Time(s)	(ν, r)	Time(s)
Iris	100±0.00	100	100±0.00	100	100±0.00	100	100±0.00	100
(150 × 4)	0.1	0.007	0.3	0.549	(0.1,0.5)	0.020	(0.4,4)	0.041
Pima	71.36±2.44	71.51	72.71±0.78	69.10	66.71±0.19	65.67	69.49±1.91	65.35
(768 × 8)	0.4	0.030	0.5	0.870	(0.6,1024)	0.100	(0.8,512)	0.495
Heart	79.25±3.75	79.67	80.83±3.28	80.59	81.25±2.10	80.97	82.08±1.64	81.90
(270 × 13)	0.3	0.038	0.4	0.214	(0.9,128)	0.094	(0.7,0.5)	0.287
Australian	86.52±1.36	86.44	87.01±0.82	87.03	90.06±0.57	89.94	94.72±0.37	94.60
(690 × 14)	0.7	0.069	0.3	0.869	(0.1,32)	0.221	(0.1,0.25)	0.878
Lung	70.58±4.8	70.14	67.64±4.15	63.25	77.64±4.92	77.09	87.06±4.92	82.81
(23 × 56)	0.3	0.008	0.6	0.398	(0.9,256)	0.015	(0.1,512)	0.053
LettersAB	98.83±0.15	98.82	97.45±0.19	97.45	99.38±0.15	99.38	99.07±0.77	99.07
(1555 × 16)	0.2	0.139	0.2	3.909	(0.3,8)	0.672	(0.1,16)	7.461

TABLE 4. Pairs taken from MNIST, ORL and YALE datasets.

Database	Samples	Classes	Cropped Pixels	Pairs Selected
MNIST	390	10	20 × 16	<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;">(0, 3) </div> <div style="text-align: center;">(1, 7) </div> <div style="text-align: center;">(4, 5) </div> <div style="text-align: center;">(6, 9) </div> <div style="text-align: center;">(2, 8) </div> </div>
ORL1 ORL2	400	40	32 × 32 64 × 64	<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;">(1, 3) </div> <div style="text-align: center;">(5, 30) </div> <div style="text-align: center;">(20, 31) </div> <div style="text-align: center;">(13, 14) </div> </div>
Yale	165	15	100 × 100	<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;">(1, 11) </div> <div style="text-align: center;">(2, 9) </div> <div style="text-align: center;">(4, 6) </div> <div style="text-align: center;">(8, 13) </div> </div>

4.3. Experiments on Six Vector-Based Datasets. Similarly, we conduct experiments on six vector-based datasets and the results are shown in Table 3. In general, our $NL\nu$ -TSTM takes 4 out of 6 better in the four algorithms from the aspect of testing accuracy. From the perspective of Gmeans, the proposed $NL\nu$ -TSTM takes 4 out of 6 better in the four algorithms; 2 out of 6 slightly worse than the other three. In addition, observing the averaged time on the six datasets, we find that our $NL\nu$ -TSTM takes more time compared with other three algorithms. The main reason is that the $NL\nu$ -TSTM adopts alternating projection method which needs more time to obtain the optimal solution during the alternative iteration.

Overall, the proposed NL ν -TSTM can handle vector-based datasets effectively and directly, and has comparable classification ability with NL ν -TSVM.

TABLE 5. Pairs taken from MNIST, ORL and YALE datasets.

Dataset	Metrics	NL ν -TSVM	NL ν -TSTM	NL ν -TSVM	NL ν -TSTM	NL ν -TSVM	NL ν -TSTM	NL ν -TSVM	NL ν -TSTM
MNIST	Num	2		4		6		8	
(0,3)	Acc	96.21±3.36	98.11±0.74	98.85±1.19	98.28±1.56	99.39±0.82	98.48±2.14	99.35±0.88	99.03±0.88
	Gmeans	96.61	98.11	98.9	98.29	99.41	98.47	99.37	99.03
(1,7)	Acc	95.40±2.04	97.29±0	95.42±3.25	97.14±0	95.75±2.71	96.97±0	96.77±3.22	97.09±0.72
	Gmeans	95.63	97.26	95.88	97.1	95.88	96.97	96.88	97.08
(4,5)	Acc	95.67±1.48	89.73±1.81	96.85±3.09	96.86±1.56	98.18±1.65	97.88±2.29	98.06±1.77	99.03±0.88
	Gmeans	95.89	89.45	96.93	96.85	98.21	97.88	98.06	99.03
(6,9)	Acc	98.92±0.60	98.38±0.60	99.43±0.78	99.43±0.78	99.09±0.83	99.39±0.82	99.03±0.88	99.68±0.72
	Gmeans	98.94	98.37	99.44	99.43	99.11	99.39	99.05	99.68
(2,8)	Acc	81.35±7.78	80.54±1.21	86.57±2.96	83.43±2.96	91.51±2.29	88.48±3.14	90±2.65	88.06±3.34
	Gmeans	82.65	79.57	88.34	82.39	91.93	88.43	91.02	87.75
ORL1	Num	1		2		3		4	
(1,3)	Acc	82.22±9.93	88.89±16.2	91.25±7.12	93.75±7.65	95.71±6.38	95.71±3.91	98.33±3.72	100±0.00
	Gmeans	85.35	88.78	92.93	93.54	96.52	95.62	98.56	100
(20,31)	Acc	78.88±12.0	88.88±12.4	83.75±12.2	100±0.00	88.57±16.44	100±0.00	96.67±7.45	100±0.00
	Gmeans	85.95	88.19	88.75	100	92.82	100	97.46	100
(5,30)	Acc	94.44±5.55	90±2.48	88.75±15.6	96.25±5.59	100±0.00	100±0.00	96.66±4.56	100±0.00
	Gmeans	95.35	89.66	92.54	96.24	100	100	97.1	100
(13,14)	Acc	91.11±8.42	95.56±4.64	91.25±10.5	97.5±3.42	98.57±3.19	97.14±3.91	98.33±3.72	100±0.00
	Gmeans	93.16	95.45	93.33	97.5	98.74	97.1	98.56	100
ORL2	Num	1		2		3		4	
(1,3)	Acc	81.11±9.29	87.78±16.9	91.25±7.12	90.0±5.59	97.14±3.91	100±0.00	98.33±3.72	100±0.00
	Gmeans	84.61	87.43	92.93	89.96	97.5	100	98.56	100
(20,31)	Acc	80±11.52	88.88±12.4	82.50±12.0	100±0.00	87.14±15.48	100±0.00	95±11.18	100±0.00
	Gmeans	86.48	88.19	87.99	100	91.58	100	96.61	100
(5,30)	Acc	95.56±4.64	87.77±2.48	88.75±15.6	95±6.84	98.57±3.19	100±0.00	96.66±4.56	100±0.00
	Gmeans	96.1	87.21	92.54	94.86	98.74	100	97.1	100
(13,14)	Acc	93.33±7.24	95.55±4.64	91.25±10.5	96.25±5.59	97.14±6.38	97.14±3.91	98.33±3.72	100±0.00
	Gmeans	94.62	95.52	93.33	96.17	97.75	97.1	98.56	100
Yale	Num	1		2		3		4	
(1,11)	Acc	96.0±2.23	100±0.00	92.22±7.45	100±0.00	87.5±7.65	100±0.00	94.28±3.19	100±0.00
	Gmeans	96.29	100	93.83	100	90.38	100	95	100
(2,9)	Acc	87.0±6.71	80.0±5.0	86.66±10.8	90±2.48	88.75±6.84	91.25±5.6	83.07±6.43	92.86±5.05
	Gmeans	87.92	79.89	87.68	89.99	91.15	91.03	87.17	92.76
(4,6)	Acc	85±14.57	84.0±17.81	88.89±5.56	97.78±4.97	87.5±0	100±0.00	87.14±3.19	98.57±3.19
	Gmeans	89.12	82.46	91.17	97.75	89.44	100	89.28	98.56
(8,13)	Acc	97.0±2.73	87.0±5.70	93.33±4.65	100±0.00	92.5±2.79	100±0.00	90±8.14	100±0.00
	Gmeans	97.26	86.95	94.29	100	93.33	100	91.95	100

5. Experiments on Tensor-based Datasets. In order to evaluate the ability of NL ν -TSTM, we choose two tensor-based databases, i.e. MNIST² and Face³. We choose ORL and Yale datasets from Face database. All features of each picture are scaled to [0,1]. We choose 5 pairs from MNIST dataset referring to their characters, we choose pairs from each dataset. In each pair, we choose 2, 4, 6 or 8 images from each category as training samples and the rest for prediction. Similarly, we choose 4 pairs for ORL1, ORL2 and Yale dataset, respectively. The detailed information is shown in Table 4. The results of NL ν -TSTM and NL ν -TSVM are summarized in Table 5.

The results show that the testing accuracies and Gmeans of NL ν -TSVM and NL ν -TSTM arise as the increase of training number. The classification accuracy of NL ν -TSTM is higher than NL ν -TSVM in most cases. The possible reason is that the vectorization in NL ν -TSVM destroyed the structural information and caused data correlation

²<http://www.cs.nyu.edu/~roweis/data.html>

³<http://www.uk.research.att.com/facedatabase.html>

damage, thus leading to poor classification ability on tensor-based datasets. While our proposed $NL\nu$ -TSTM can retain more structural information and performs better on HDS3 problem. That also verified the effectiveness and rationality of our algorithm. Besides, the averaged Gmeans of $NL\nu$ -TSTM is higher than $NL\nu$ -TSVM in most cases. That indicates our $NL\nu$ -TSTM can overcome the overfitting problem to a large extent.

6. Conclusions. We propose a new tensor-kernel based algorithm named the $NL\nu$ -TSTM. Compared with ν -TSVM, it deals with tensor data directly and utilizes more data structural information. Besides, it solves two smaller-sized QPPs to reduce its computational complexity in each iteration. Moreover, it can avoid the overfitting problem to some extent. As respected, the computational experiments testified its superiorities. However, the iteration process of obtaining optimal solutions is time consuming. In the future, the possible research direction is to design a fast-solving method for $NL\nu$ -TSTM.

Acknowledgment. This work is partially supported by the China Agriculture Research System (CARS-29), National Natural Science Foundation of China (No.11671010) and the Key Laboratory of Viticulture and Enology, Ministry of Agriculture.

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

REFERENCES

- [1] V. N. Vapnik. *The nature of statistical learning theory*. Berlin: Springer, 1995.
- [2] Jayadeva, R. Khemchandani and S. Chandra, twin support vector machines for pattern classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.29, no. 5, pp. 905-910, 2007.
- [3] X. Peng, A ν -twin support vector machine (ν -TSVM) classifier and its geometric algorithms, *Information Sciences*, vol. 180, pp. 3863-3875, 2010.
- [4] X. Peng, Least squares twin support vector hypersphere (LS-TSVH) for pattern recognition, *Expert Systems with Applications*, vol. 37, no. 12, pp. 8371-8378, 2010.
- [5] Y. Xu, L. Wang and P. Zhong, A rough margin-based ν -twin support vector machine, *Neural Computing and Applications*, vol. 21, pp. 1307-1317, 2012.
- [6] H. Wang and Z. Zhou, An improved rough margin-based ν -twin bounded support vector machine, *Knowledge-Based Systems*, vol. 128, pp. 125-138, 2017.
- [7] R. Green and L. Guan, Quantifying and recognizing human movement patterns from monocular video images-part II: applications to biometrics, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 2, pp. 191-198, 2004.
- [8] D. Tao, X. Li, X. Wu, and S. J. Maybank, Supervised tensor learning, *Knowledge and Information Systems*, vol. 13, no. 1, pp. 1-42, 2007.
- [9] D. Cai, X. Hei, and J. Han. *Learning with Tensor Representation*. 2006.
- [10] R. Khemchandani, A. Karpatne, and S. Chandra, Proximal support tensor machines, *International Journal of Machine Learning and Cybernetics*, vol. 4, no. 6, pp. 703-712, 2013.
- [11] I. Kotsia, W. W. Guo, and I. Patras, Higher rank support tensor machines for visual recognition, *Pattern Recognition*, vol. 45, no. 12, pp. 4192-4203, 2012.
- [12] Z. Hao, L. He, B. Chen and X. Yang, A linear support higher-order tensor machine for classification, *IEEE Transactions on Image Process*, vol. 22, no. 7, pp. 2911-2920, 2013.
- [13] H. Xu, L. Fan, and X. Gao, TBSTM: A novel and fast nonlinear classification method for image data, *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 29, no. 8, 2015.
- [14] C. Gao and X. Wu, Kernel support tensor regression, *Procedia Engineering*, vol. 29, pp. 3986-3990, 2012.
- [15] S. Park. Multifactor Analysis for fMRI Brain Image Classification by Subject and Motor Task. *2011 Electrical and Computer Engineering Technical Report*, Carnegie Mellon University: 2011.
- [16] L. F. He, X. N. Kongy, and P. S. Yu et al. , DuSK: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages, *Matrix*, vol. 3, no. 1, pp. 2, 2014.
- [17] X. Z. Gao, L. Fan, and H. Xu, NLS-TSTM: A novel and fast nonlinear image classification method, *WSEAS Transactions on Mathematics*, vol. 13, no. 1, pp. 626-635, 2014.

- [18] Y. Chen, K. Wang, and P. Zhong, One-class support tensor machine, *Knowledge-Based Systems*, vol. 96, pp. 14-28, 2016.
- [19] H. Han, Analyzing support vector machine overfitting on microarray data, *Journal of Bioinformatics and Computational Biology*, vol. 8590, pp. 148-156, 2014.