A \( \nu \)-Twin Support Tensor Machine*  

Huiru Wang\(^{1,a}\), Haoyu Wu\(^{1,b}\), Zhijian Zhou\(^{1,c,}\)*  

\(^{1}\)College of Science, China Agricultural University, Beijing 100083, China  

\(^{a}\)wanghuiru529@163.com, \(^{b}\)whyshrha@gmail.com, \(^{c}\)zhijianzh@163.com  

*Corresponding author  

**Keywords:** Support vector machine; \( \nu \)-twin support vector machine; Tensor learning; Classification problem; High dimensional and small sample size problem  

**Abstract.** The traditional vector-based algorithms, such as Support Vector Machine (SVM) and Twin Support Vector Machine (TSVM), have many limitations especially when tensor is considered as input matrix. In this paper, we proposed a novel algorithm with a tensor-based classification paradigm to utilize the structural information. The new proposed algorithm is called \( \nu \)-Twin Support Tensor Machine (\( \nu \)-TSTM), which is an extension of \( \nu \)-Twin Support Vector Machine (\( \nu \)-TSVM). Similarly, \( \nu \)-TSTM solves a pair of smaller-sized Quadratic Programming Problems (QPPs). It reduces the computational complexity substantially. Besides, we formulate \( \nu \)-TSTM, which separates samples in the tensor space with two non-parallel hyperplanes, and the pair of parameters (\( \nu \)) have theoretical interpretation which are used to control the bounds of the fractions of support tensors and the error margins. What’s more, the structure information of data is retained by the direct use of tensor representation. The proposed \( \nu \)-TSTM can preferably overcome overfitting problem and deal with big data while most vector-based algorithms could hardly compare. In addition, it has better performances on high dimensional and small-sample-size (S3) problem. The efficiency and superiority of the proposed method are demonstrated by experiments on various datasets.  

**Introduction**  

Pattern recognition problem (classification problem) is a vital branch of machine learning [1]. Support Vector Machine (SVM) [2], as one of the effective and promising classifiers in machine learning today, has been successfully applied in many fields, such as web image annotation and disease diagnosis [3; 4].  

However, one of the challenges for SVM is the high computational complexity. Therefore, many methods are proposed to solve this problem, the most famous one is Twin Support Vector Machine (TSVM) which is introduced by Jayadeva [5]. TSVM seeks two nonparallel proximal hyperplanes by solving two smaller-sized QPPs while SVM solves a larger one, which makes it works faster than SVM. Later, many variants of TSVM have been proposed, such as Least Squares Twin Support Vector Machine (LS-TSVM) [6] and \( \nu \)-Twin Support Vector Machine (\( \nu \)-TSVM) [7]. While all these methods are based on vector space, it may fail when tensor is considered as input data. In fact, lots of the original objects are represented as tensor form. For instance, a gray face image is represented by the second order tensor [8]. Although there are methods converting tensor directly into vector, it may cause structural information lose and data correlation damage [9]. What’s more, it often leads to overfitting, curse of dimensionality problem and small sample size (S3) problem.  

In order to utilize the structural information and solve the aforementioned problems, many algorithms based on tensor space are proposed for pattern classification. Tao and Cai proposed  

*This work is supported by Chinese Universities Scientific Fund (No.2016LX002)*
Support Tensor Machine (STM) [10] which deals with input image directly without vectorization. In addition, the experimental results also verified that the classification accuracy of STM is superior to that of the traditional SVM. Later, more and more researches concentrate on the study from the vector-based algorithm to tensor-based classification approaches, and most have gained good performance and application. Linear C-SVM and $\nu$-SVM [11] are extended to tensor based form C-STM and $\nu$-STM by Tao et al. [9]. Chen et al. addressed one-class classification problem with tensor-based maximal margin classification paradigm and formulated One-Class Support Tensor Machine (OCSTM) [12]. In order to utilize the structural information present in high dimensional features, tensor based algorithm Fuzzy Least Squares Support Tensor Machine (FLSSTM) is proposed by Zhang et al. [13] Kotsia et al. formulated the higher rank STMs in which the separating hyperplane is defined by the parameters and constrained to be the sum of rank one tensors [14]. Zhang et al. generalized vector-based TSVM to tensor-based algorithm TSTM and the proposed algorithm were applied for micro calcification clusters detection [15]. Compared with TSVM, tensor-based TSTM substantially reduced the overfitting problem. Zhao et al. extended LSTSVM to the general tensor forms Least Squares Twin Support Tensor Machine (LSTSTM) [16].

In this paper, motivated by these aforementioned works, we propose a new algorithm based on tensor space, called $\nu$-Twin Support Tensor Machine ($\nu$-TSTM) by introducing additional variables $\rho$ and new parameters $\nu$. Similar to $\nu$-TSVM, the parameters $\nu$ in $\nu$-TSTM have a better theoretical interpretation. And the pair of parameters ($\nu$) is used to control the bounds of the fractions of support tensors and the error margins. The main idea of $\nu$-TSTM is to find two non-parallel hyperplane in tensor space and solve a pair of smaller-sized QPPs, which reduce its computational complexity substantially compared with STM. And the direct use of tensor representation reserved the structure of data information. What’s more, the proposed $\nu$-TSTM can avoid overfitting problem to a large extent and more suit for S3 problem. We give a detail experiment and analysis on Australian dataset. The efficiency and superiority of the proposed method are also demonstrated by experiments on both vector and tensor datasets.

The remainder of the paper is organized as follows. In Section Related Work we give a brief review of STM and $\nu$-TSVM. In the following Section we introduce our algorithm $\nu$-TSTM and its theoretical interpretation. The experimental results on vector-based datasets and tensor-based datasets are presented later. Finally, we have some discussions and conclusions.

**Related Work**

In this section, we make a brief overview of STM and $\nu$-TSVM.

STM [10] is a method to solve a tensor generalization of SVM in the tensor space. Consider the classification problem for tensor generalization. Suppose the training data $\{x_i, y_i\} (i = 1, 2, \cdots, l)$, where $x_i \in \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2}$ is a sample point in 2nd-order tensor space; $y_i \in \{-1, 1\}$ is the label of $x_i$; $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$ are two vector spaces. The linear decision function of STM in the tensor space can be described as follows:

$$f(x) = \text{sgn} (u^T x v + b), u \in \mathbb{R}^{n_1}, v \in \mathbb{R}^{n_2}. \quad (1)$$

The decision function (1) can be acquired by solving the optimization problem, which can be introduced as:
In order to solve the above problem (2), some methods have been introduced in [9; 17], which are simple and effective. Firstly, give an initial value \( u \), and let \( \beta_i = \|u\|^2 \) and \( \mathbf{x}_i = \mathbf{X}_i^T \mathbf{u} \). Hence, the optimization problem is formulated as follows:

\[
\begin{align*}
\min_{\mathbf{v} \in \mathbb{R}^n, \beta_i \in \mathbb{R}, \xi_i \in \mathbb{R}_+} & \quad \frac{1}{2} \|\mathbf{v}\|^2 + C \sum_{i=1}^l \xi_i \\
\text{s.t.} & \quad y_i (\mathbf{u}^T \mathbf{x}_i + b) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0, i = 1, \ldots, l.
\end{align*}
\]

(2)

Obviously, the optimization problem (3) is similar to a standard SVM problem. So it can be solved by any calculation method of SVM. After the value \( \mathbf{v} \) is obtained, let \( \beta_2 = \|\mathbf{v}\|^2 \) and \( \tilde{\mathbf{x}}_i = \mathbf{X}_i \mathbf{v} \). The formulation is changed to another optimization problem:

\[
\begin{align*}
\min_{\mathbf{u} \in \mathbb{R}^n, \beta_2 \in \mathbb{R}, \xi_i \in \mathbb{R}_+} & \quad \frac{1}{2} \beta_2 \|\mathbf{u}\|^2 + C \sum_{i=1}^l \xi_i \\
\text{s.t.} & \quad y_i (\mathbf{u}^T \tilde{\mathbf{x}}_i + b) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0, i = 1, \ldots, l.
\end{align*}
\]

(3)

Same like problem (3), the solution of problem (4) can be obtained by the methods of SVM. Then the value \( \mathbf{u} \) and \( b \) is gained. Go on to alternate iteration likes above steps until the algorithm meets the stop condition [9; 17]. After the value of \( \mathbf{u}, \mathbf{v} \) and \( b \) obtained, we can get the separating hyperplane \( f(\mathbf{x}_i) = \langle \mathbf{X}_i, \mathbf{u} \mathbf{v}^T \rangle + b = \mathbf{u}^T \mathbf{X}_i \mathbf{v} + b \).

\( \nu \)-Twin Support Vector Machine (\( \nu \)-TSVM). \( \nu \)-Twin Support Vector Machine, which was developed by Peng [7], is an improved algorithm of Twin Support Vector Machine. As the same as the classical SVM, the trade-off factors \( c_1 \) and \( c_2 \) in the TSVM do not own any other theoretical interpretation except for controlling the ratios of empirical risks. Through a pair of parameters \( \nu_1 \) and \( \nu_2 \), the bounds of the fractions of the support vectors and the margin errors can be controlled.

For linear case, \( \nu \)-TSVM seeks the following pair of nonparallel positive and negative hyperplanes:

\[
\langle \mathbf{w}_+, \mathbf{x} \rangle + b_+ = 0 \quad \text{and} \quad \langle \mathbf{w}_-, \mathbf{x} \rangle + b_- = 0.
\]

(5)

Such that each hyperplane is closer to the samples of one of the two classes and is as far as possible from those of the other class.
The solution of $\nu$-TSVM is acquired by solving the following optimization problems.

$$\min_{w,b,\xi} \frac{1}{2} \|Aw + e_b\|^2 - \nu_1 \rho_1 + \frac{1}{l} e^T_1 \xi_-$$

s.t. $-(Bw + e_b) \geq \rho_1 e - \xi_- \quad (6)$

$$\rho_1 \geq 0, \xi_- \geq 0$$

$$\min_{w,b,\xi} \frac{1}{2} \|Bw + e_b\|^2 - \nu_2 \rho_2 + \frac{1}{l} e^T_2 \xi_+$$

s.t. $Aw + e_b \geq \rho_2 e - \xi_+$

$$\rho_2 \geq 0, \xi_+ \geq 0$$

By introducing Lagrange multipliers $\alpha$ and $\beta$, the dual problems (8) and (9) are obtained as follows.

$$\min_{\alpha} \frac{1}{2} \alpha^T G (H^T H)^{-1} G^T \alpha$$

s.t. $0 \leq \alpha \leq \frac{1}{l} e_-$

$$\quad e^T \alpha \geq v_1$$

$$\min_{\beta} \frac{1}{2} \beta^T P (Q^T Q)^{-1} P^T \beta$$

s.t. $0 \leq \beta \leq \frac{1}{l} e_+$

$$\quad e^T \beta \geq v_2$$

Where $G = [B \ e_-], H = [A \ e_-], P = [A \ e_+]$ and $Q = [B \ e_+]$. Then we have

$$[w_+ b_+]^T = -(H^T H)^{-1} G^T \alpha, \quad [w_- b_-]^T = (Q^T Q)^{-1} P^T \beta.$$  

(10)

So, if there is a new point $x$, the decision function is

$$\text{class } k = \arg \min_{k=+,-} \frac{\langle w^{(k)}, x \rangle + b^{(k)} \rangle}{\|w^{(k)}\|}.$$

(11)

$\nu$-Twin Support Tensor Machine ($\nu$-TSTM)

In this section, we propose a novel algorithm named $\nu$-Twin Support Tensor Machine, which is generalized based on $\nu$-TSVM from the vector space to the tensor space.

For binary classification of tensor data, input the training dataset:

$$T = \{(X_1, +1), (X_2, +1), \ldots, (X_p, +1), (X_{p+1}, -1), \ldots, (X_{p+q}, -1)\},$$

(12)

where $X_i \in \mathbb{R}^n \otimes \mathbb{R}^n$ represents the second-order tensor (matrix). $\mathbb{R}^n$ and $\mathbb{R}^n$ are two vector spaces. Similarly, $\nu$-TSVM seeks the following pair of nonparallel positive and negative hyperplanes:

$$\langle W_+, X \rangle + b_+ = 0 \quad \text{and} \quad \langle W_-, X \rangle + b_- = 0.$$  

(13)
\textbf{\( \nu\)-TSTM and Algorithm.} As similar as \( \nu\)-TSVM, the model of algorithm can be denoted as follows:

\[
\min_{w, b, \nu, \rho, \xi} \frac{1}{2} \sum_{i=1}^{d} ((w_i, x_i) + b_i)^2 - \nu \rho + \frac{1}{q} \sum_{j=1}^{q} \xi_j
\]

\text{s.t.} \quad -(w_i, x_j) + b_i \geq \rho - \xi_j, \quad \rho_i \geq 0, \quad \xi_j \geq 0, \quad j = 1, \ldots, q

(14)

\[
\min_{w, b, \nu, \rho, \xi} \frac{1}{2} \sum_{i=1}^{p} ((w_i, x_i) + b_i)^2 - \nu \rho + \frac{1}{p} \sum_{i=1}^{p} \xi_i
\]

\text{s.t.} \quad (w_i, x_i) + b_i \geq \rho - \xi_i, \quad \rho_i \geq 0, \quad \xi_i \geq 0, \quad i = 1, \ldots, p

(15)

Where \( w_i, x_i \in \mathbb{R}^n \otimes \mathbb{R}^n \), \( b_i, b_+ \in \mathbb{R} \). \( \xi_i, \xi_j \) are slack variables. \( \nu_1, \nu_2 \) are new parameters. \( \rho_+, \rho_- \) are additional variables. It is note that for all \( \xi_j = 0, j = 1, \ldots, q \) (or \( \xi_i = 0, i = 1, \ldots, p \)), the negative (or positive) samples are separated by the positive (or negative) hyperplane, with the margin \( \rho_+ / \|W_+\|^2 \) (or \( \rho_- / \|W_-\|^2 \)).

For solving the optimization problem (14) and (15), the method of Rank-one was introduced in [10]. \( W_+, W_- \) are restricted to be Rank-one, then they can be decomposed like \( W_+ = uv^T, W_- = \bar{u}\bar{v}^T \).

According to decomposition operations and tensor multiplication [18]:

\[
\langle W_+, X \rangle = \langle uv^T, X \rangle = X \times u \times v = u^T X v,
\]

(16)

\[
\langle W_-, X \rangle = \langle \bar{u}\bar{v}^T, X \rangle = X \times \bar{u} \times \bar{v} = \bar{u}^T \bar{X} \bar{v}.
\]

(17)

To make the optimization problem (14) and (15) more simple and easy to be solved, we transform the positive training dataset and negative training dataset into third-order tensors \( A \) and \( B \) respectively. The Rank-one formulation of (14) and (15) can be denoted as follows:

\[
\min_{u, v, b, \rho, \xi} \frac{1}{2} \| B \times u \times v + e \| - \nu \rho + \frac{1}{q} \| e_j \xi \|
\]

\text{s.t.} \quad -(B \times u \times v + e) \geq \rho e - \xi, \quad \rho_+ \geq 0, \quad \xi_+ \geq 0

(18)

\[
\min_{\bar{u}, \bar{v}, b, \rho, \xi} \frac{1}{2} \| B \times \bar{u} \times \bar{v} + \bar{e} \| - \nu \rho + \frac{1}{p} \| \bar{e}_i \xi \|
\]

\text{s.t.} \quad (A \times \bar{u} \times \bar{v} + \bar{e}) \geq \rho e_+ - \xi_+, \quad \rho_- \geq 0, \quad \xi_- \geq 0.

(19)

Where \( \xi_+ = (\xi_1, \xi_2, \ldots, \xi_p) \), \( \xi_- = (\xi_1, \xi_2, \ldots, \xi_q) \). The symbols of \( e_+, e_- \) are the unit vectors which match with \( \xi_+, \xi_- \) respectively.

On account of the similar structure of the optimization problem (18) and (19), we only describe the solution of the problem (18). We introduce positive Lagrange multipliers \( \alpha, \beta \) and \( \lambda \). Then the Lagrange formulation for problem (18) is
The Karush-Kuhn-Tucker (K.K.T) conditions [19] are as follows:

\[
\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = (A \times \mathbf{v})(A \times \mathbf{u} + \mathbf{e}_b) + (B \times \mathbf{v})\mathbf{a} = 0 \quad (21)
\]

\[
\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = (A \times \mathbf{u})(A \times \mathbf{u} + \mathbf{e}_b) + (B \times \mathbf{u})\mathbf{a} = 0 \quad (22)
\]

\[
\frac{\partial \mathcal{L}}{\partial b_i} = \mathbf{e}_i^T (A \times \mathbf{v} + \mathbf{e}_b) + \mathbf{e}_i^T \mathbf{a} = 0 \quad (23)
\]

\[
\frac{\partial \mathcal{L}}{\partial \rho_+} = \mathbf{e}_+ - \mathbf{v} - s = 0 \quad (24)
\]

\[
\frac{\partial \mathcal{L}}{\partial \xi_-} = \mathbf{e}_- / q - \mathbf{a} - \mathbf{r} = 0. \quad (25)
\]

From the Eq. (21) and Eq. (22), we get that \( \mathbf{u} \) and \( \mathbf{v} \) are interdependent. The traditional methods cannot solve the problems. So the Alternate Iterating Algorithm [10; 17] is applied to the problem (18).

In the first place, we initialize \( \mathbf{u}_0 \). Let \( \mathbf{A}_i = A \times \mathbf{u} \) and \( \mathbf{B}_i = B \times \mathbf{u} \). Then the optimization problem (18) can be transformed as follows:

\[
\min_{\mathbf{v}, \mathbf{b}_i, \mathbf{u}_i} \frac{1}{2} \| \mathbf{A}_i \mathbf{v} + \mathbf{e}_b \| - \mathbf{v}_i \mathbf{\rho}_+ + \frac{1}{q} \mathbf{e}_+^T \xi_- = 0
\]

\[
\text{s.t.} \quad -(\mathbf{B}_i \mathbf{v} + \mathbf{e}_b) \geq \mathbf{\rho}_+ \mathbf{e}_- - \xi_- \quad (26)
\]

\[
\mathbf{\rho}_+ \geq 0, \quad \xi_- \geq 0.
\]

Based on the Lagrange and K.K.T. conditions, we can also get the result of problem (26) in the above.

\[
\mathbf{\eta}_i = -(\mathbf{H}_i \mathbf{H}_i)^{-1} \mathbf{G}_i^T \mathbf{a}. \quad (27)
\]

Where \( \mathbf{H}_i = [\mathbf{A}_i \mathbf{e}_i], \mathbf{G}_i = [\mathbf{B}_i \mathbf{e}_i], \) and the augmented vector \( \mathbf{\eta}_i = [\mathbf{v} \mathbf{b}_i]^T \).

Note that \( \mathbf{H}_i^T \mathbf{H}_i \) is always positive semidefinite, but it may be ill-conditioned in some situations. Through introducing a regularization term [20] \( \varepsilon \mathbf{I}, \varepsilon > 0 \), avoid possible ill-conditioning of \( \mathbf{H}_i^T \mathbf{H}_i \). Hence, (27) gets modified to

\[
\mathbf{\eta}_i = -(\mathbf{H}_i^T \mathbf{H}_i + \varepsilon \mathbf{I})^{-1} \mathbf{G}_i^T \mathbf{a}. \quad (28)
\]

Then the Wolfe dual of optimization problem(26) can be obtained:

\[
\max_{\mathbf{a}} -\frac{1}{2} \mathbf{a}^T \mathbf{G}_i (\mathbf{H}_i^T \mathbf{H}_i)^{-1} \mathbf{G}_i^T \mathbf{a}
\]

\[
\text{s.t.} \quad 0 \leq \mathbf{a} \leq \mathbf{e}_- / q
\]

\[
\mathbf{e}_+^T \mathbf{a} \geq \mathbf{v}_i. \quad (29)
\]

We can get \( \mathbf{\eta}_i = [\mathbf{v} \mathbf{b}_i]^T \) by solving (29) and the Eq. (28).
Secondly, when the value \( v \) is acquired, we further solve \( u \) as above. Let \( A_2 = A \times v \) and \( B_2 = B \times v \). According to (18), \( u \) can be computed by the following quadratic programming problem (30), which is similar to problem (26).

\[
\begin{align*}
\min_{\bar{u}, \bar{b}, \rho, \xi} & \quad \frac{1}{2}\|u^T A_2 + e, b, \|^2 - v, \xi - \frac{1}{q} - e^T \xi \\
\text{s.t.} & \quad -(u^T B_2 + e, b, ) \geq \rho, e, - \xi \\
& \quad \rho, \geq 0, \quad \xi, \geq 0 \\
\end{align*}
\]

(30)

The Wolfe dual of optimization problem (30) can be represented as well.

\[
\begin{align*}
\max_{\bar{u}} & \quad -\frac{1}{2}\tilde{a}^T G_2 (H_2^T H_2)^{-1} G_2^T \tilde{a} \\
\text{s.t.} & \quad 0 \leq \tilde{a} \leq e, /q \\
& \quad e^T \tilde{a} \geq \nu, \\
\end{align*}
\]

(31)

Therefore, in the similar method we can acquire

\[
\eta, = -(H_2^T H_2)^{-1} G_2^T \tilde{a},
\]

(32)

Where \( H_2 = [B_2 e,], \ G_2 = [A_2 e,] \) and \( \eta, = [u b,]^T \).

Thus, \( u \) and \( v \) can be obtained by iteratively solving the optimization problems (26) and (30).

Next we only describe the positive hyperplane algorithm, the negative hyperplane algorithm is in the similar way.

**The Algorithm of \( \nu \)-TSTM**

**Inputs:** the value \( \nu, \) and dataset

\[
T = \{(X_1, +1),(X_2, +1),\ldots,(X_p, +1),(X_{p+1}, -1),\ldots,(X_{p+q}, -1)\}, X, \in \mathbb{R}^n \otimes \mathbb{R}^n .
\]

**Outputs:** the positive hyperplane \( \langle W, X, \rangle + b, = 0 \)

**Step1:** Initialize the \( u = (1,\ldots,1)^T \).

**Step2:** Let \( A_1 = A \times u, \ B_1 = B \times u \), then \( v, b, \) can be obtained by solving the problems (26)

**Step3:** After acquired \( v \) in Step2, let \( A_2 = A \times v \) and \( B_2 = B \times v \), then \( u, b, \) can be obtained by solving the problems (30).

**Step4:** Compute \( u \) and \( v \) by the Alternate Iterating. If the following conditions: \( \|u, -u,-\| \leq \varepsilon, \) \( \|v, -v,-\| \leq \varepsilon \) and \( \|b, -b,-\| \leq \varepsilon \) (where \( \varepsilon \approx 0 \) ) are satisfied simultaneously or the iteration times is more than the maximum number of iteration, the iteration will be terminated.

**Theoretical Interpretation of \( \nu \)-TSTM.** In this subsection, we still use positive hyperplane to interpret. After solving (18), we can achieve its solution \( a^* = (\alpha,^* , \alpha,^2 , \ldots , \alpha,^q )^T \). The different values \( \alpha,^j \) of are corresponding to the different positions of negative samples. Hence, we drive the following proposition.
**Proposition 1.** The negative samples locating in different positions have different values of $\alpha_j^*$ and they can be divided into three cases.

1. If $\alpha_j^* = 0$, then the negative samples lie under the hyperplane $\langle W_s, X \rangle + b_s = -\rho_s$. They are the negative samples which are correctly classified.

2. If $0 < \alpha_j^* < 1/q$, which means $\xi_j = 0$ and the negative samples lie on the hyperplane $\langle W_s, X \rangle + b_s = -\rho_s$. They are the support tensors of the negative classes.

3. If $\alpha_j^* = 1/q$, then the samples lie upper the hyperplane $\langle W_s, X \rangle + b_s = -\rho_s$. They are usually the outliers or noises of the negative classes.

And according to K.K.T. condition, we can calculate the parameter $\rho_s$:

$$\rho_s = -\frac{1}{n_{sv}} \sum_{j=1}^{n_{sv}} (\langle W_s, X_j \rangle + b_s),$$

where $n_{sv}$ denotes the number of negative support tensors.

And the fraction of negative margin error is $R_{mp}^* = (1/q) \left| \left\{ j \mid \langle W_s, X_j \rangle + b_s > -\rho_s, j = 1, \ldots, q \right\} \right|$. And the following proposition gives the theoretical interpretation of $\nu_i$:

**Proposition 2.** Suppose we run problem (18) with $(p+q)$ samples on dataset, acquiring the result that $\rho_s > 0$. Then

1. $\nu_i$ is an upper bound on the fraction of negative margin errors.
2. $\nu_i$ is a lower bound on the fraction of negative support tensors.

**Numerical Experiments**

In this section we present the experimental results on some problems from the UCI Repository of machine learning databases [21]. Firstly, we evaluate the proposed $\nu$-TSTM algorithm on Australian dataset for a trial of detailed discussion. Then we give an overall comparison on all vector-based datasets. All algorithms are written in Mat lab 2010a using M-file. These programs are operated on 3.60GHz Inter Core i3-4160 Duo CPU with 4.0GB RAM. We use 5-fold cross-validation on the training set to find the best parameters. There are three running parameters: $C$, $\nu_1$ and $\nu_2$. The optimal parameters $C$ in SVM and STM were searched from $C = [2^{-5}, 2^{-4}, \ldots, 2^5]$. We set $\nu_1 = \nu_2 = \nu$, and the optimal parameters $\nu$ in the $\nu$-TSTM and $\nu$-TSVM were searched in the range $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. We evaluate the performance of classifiers according to testing accuracy, which is defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

where $TP$, $TN$, $FP$ and $FN$ respectively stands for numbers of points of true positive, true negative, false positive and false negative.

At the same time, we discuss the overfitting problem which comes across in the vector-based algorithm with high dimensional and S3 problem. To verify the effectiveness of dealing with overfitting problem, we use Sensitivity and Specificity to calculate, where:
\[
Sensitivity = \frac{TP}{TP + FP}; \quad \text{Specificity} = \frac{TN}{TN + FN}
\]

*Time* is the average training time of 5-fold cross-validation, the unit of *Time* is seconds(s).

**Experiments on Australian Dataset.** In this section, we do the experiments on Australian dataset, which has 383, 307 positive and negative samples respectively, with 14 attributes. And we discuss the impact of different tensor sizes, different training sizes on classification performance and overfitting problems, respectively. These experiments are described fully in next subsections.

**Experiments on Different Tensor Sizes.** In this subsection, we discuss on the impact of the size of tensor on classification performance. For a vector sample \( \mathbf{x} \in \mathbb{R}^n \), it can be converted to the form of matrix (second order tensor) \( \mathbf{X} \in \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2} \), where \( n_1 \times n_2 \approx n \). Cai [10] proposed a method to make the conversion. Here, we draw Fig.1 to show the five possible tensor sizes that converted from vector in Australian dataset. It is worthwhile to find which conversion is the best one. Our experiments record the testing accuracy on different tensor sizes and different training sizes using our proposed \( \nu \)-TSTM algorithm. The results are displayed in Table 1. The bold form is the best one. And we know that when tensor size is \( 4 \times 4 \), \( \nu \)-TSTM obtain pleasurable performance.

![Figure 1](image1.png)  
**Figure 1.** Five possible types of tensor size in Australian dataset.

![Figure 2](image2.png)  
**Figure 2.** Classification performance on Australian dataset with respect to training sample sizes.

**Table 1.** Averaged testing accuracy in different tensor sizes and different training sizes using \( \nu \)-TSTM.

<table>
<thead>
<tr>
<th>Training numbers</th>
<th>Tensor size</th>
<th>2 × 7</th>
<th>3 × 5</th>
<th>4 × 4</th>
<th>5 × 3</th>
<th>7 × 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>74.55±4.96</td>
<td>74.41±3.69</td>
<td><strong>77.84±2.64</strong></td>
<td>76.86±0.63</td>
<td>67.53±1.16</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>78.31±9.35</td>
<td>79.85±7.18</td>
<td><strong>84.84±0.11</strong></td>
<td>83.38±1.95</td>
<td>71.69±2.39</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>84.92±0.22</td>
<td>85.07±0.45</td>
<td><strong>85.16±1.01</strong></td>
<td>84.61±0.22</td>
<td>75.32±3.03</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>84.51±0.57</td>
<td>83.36±2.89</td>
<td>85.33±0.57</td>
<td><strong>85.49±0.12</strong></td>
<td>84.84±0.41</td>
<td></td>
</tr>
</tbody>
</table>

The experiments results indicate that the closer of \( n_1 \) and \( n_2 \), the better classification performance. In addition, it is necessary to consider the actual tensor forms when making the transformation. Based on these two principles, we establish the tensor sizes of all vector datasets involved in this paper, shown in Table 5.

**Experiments on Classification Performance.** In this subsection, we use four algorithms to testify the classification results with various training sample sizes, the results are in Table 2. The results
indicate that our proposed algorithm $\nu$-TSTM is outstanding on entirely different training sizes with respect to testing accuracy. Fig.2 shows the testing accuracy variation tendency respect to different training sizes. It shows that with the training number increasing, the testing accuracy mostly arises, and the method of $\nu$-TSTM we proposed performs best. Table 3 records the best parameter and the averaged training time. We get that the more numbers of training samples, the more running time needed. The order of running speed is $\nu$-TSVM>SVM>$\nu$-TSTM>STM under the same training sample. Although our proposed $\nu$-TSTM is a little bit slower than vector based SVM and $\nu$-TSVM, it works faster than tensor-based algorithm STM. The reason is that the tensor-based algorithm need to iterate.

Table 2. Averaged testing accuracy on various training sample sizes on Australian dataset.

<table>
<thead>
<tr>
<th>Number</th>
<th>SVM</th>
<th>STM</th>
<th>$\nu$-TSVM</th>
<th>$\nu$-TSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>72.77±3.02</td>
<td>71.14±2.82</td>
<td>68.00±9.42</td>
<td><strong>77.84±2.64</strong></td>
</tr>
<tr>
<td>20</td>
<td>75.07±2.54</td>
<td>74.21±3.69</td>
<td>80.15±6.22</td>
<td><strong>84.84±0.11</strong></td>
</tr>
<tr>
<td>30</td>
<td>68.03±5.89</td>
<td>72.12±5.77</td>
<td>83.24±2.47</td>
<td><strong>85.16±1.01</strong></td>
</tr>
<tr>
<td>40</td>
<td>67.37±5.96</td>
<td>71.47±2.41</td>
<td>84.54±2.29</td>
<td><strong>85.33±0.57</strong></td>
</tr>
</tbody>
</table>

Table 3. Averaged training time and best parameter on various training sample sizes on Australian dataset.

<table>
<thead>
<tr>
<th>Number</th>
<th>SVM</th>
<th>STM</th>
<th>$\nu$-TSVM</th>
<th>$\nu$-TSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>C</td>
<td>Time</td>
<td>ν</td>
</tr>
<tr>
<td>10</td>
<td>0.047</td>
<td>$2^{-4}$</td>
<td>1.482</td>
<td>$2^{3}$</td>
</tr>
<tr>
<td>20</td>
<td>0.078</td>
<td>$2^{-5}$</td>
<td>4.009</td>
<td>$2^{(-2)}$</td>
</tr>
<tr>
<td>30</td>
<td>0.109</td>
<td>$2^{-5}$</td>
<td>6.708</td>
<td>$2^{5}$</td>
</tr>
<tr>
<td>40</td>
<td>0.156</td>
<td>$2^{5}$</td>
<td>9.641</td>
<td>$2^{(-4)}$</td>
</tr>
</tbody>
</table>

Table 4. Averaged sensitivity and specificity on various training sample sizes on Australian dataset.

<table>
<thead>
<tr>
<th>Number</th>
<th>SVM</th>
<th>STM</th>
<th>$\nu$-TSVM</th>
<th>$\nu$-TSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Sensitivity</td>
<td>81.71±13.57</td>
<td>68.36±12.36</td>
<td>60.51±22.54</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>61.55±12.62</td>
<td>74.63±17.98</td>
<td><strong>77.41±13.81</strong></td>
</tr>
<tr>
<td>20</td>
<td>Sensitivity</td>
<td><strong>82.14±11.85</strong></td>
<td>70.43±11.43</td>
<td>74.76±13.49</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>66.13±14.19</td>
<td>78.97±17.41</td>
<td>86.97±5.56</td>
</tr>
<tr>
<td>30</td>
<td>Sensitivity</td>
<td>78.47±16.22</td>
<td><strong>81.39±17.24</strong></td>
<td>80.79±4.69</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>54.72±22.74</td>
<td>60.28±22.82</td>
<td>86.35±7.03</td>
</tr>
<tr>
<td>40</td>
<td>Sensitivity</td>
<td>72.53±21.84</td>
<td>77.84±1.77</td>
<td>80.87±4.71</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>60.74±29.71</td>
<td>63.29±7.79</td>
<td>89.25±3.21</td>
</tr>
</tbody>
</table>
**Experiments on Overfitting Problem.** In this subsection, we discuss the overfitting problem which encountered in the vector-based algorithm with high dimensional and S3 problems. And we still use sensitivity and specificity to verify the effectiveness of SVM, STM, $\nu$-TSVM and $\nu$-TSTM, dealing with overfitting problem. Table 4 summarizes the averaged sensitivity and specificity on various training samples sizes with the best parameter on Australian dataset. The best accuracy is in bold. The results show that $\nu$-TSTM has promising advantages on avoiding overfitting problem.

**Experiments on Various Datasets.** In this section, we do computational experiments on various kinds of datasets to verify the ability of our proposed algorithm. All these datasets are downloaded from UCI database [21]. There are 12 datasets to be used in our experiments and the description of various datasets is in Table 5. In all experiments, the 12 datasets are used to construct the binary classification problems. The first principle of choosing samples is avoiding imbalance problem. And for vector-based dataset, we transform the vector to tensor form to make $n_1$ and $n_2$ nearly as much as possible. For tensor-based dataset, we obey the actual tensor form. In a similar spirit to the experiments on Australian dataset, we focus on small training sets and compute averaged testing accuracy, sensitivity and specificity.

### Table 5. Description of various datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Number of samples P+N</th>
<th>Training-size P+N</th>
<th>Testing-size P+N</th>
<th>Attribute</th>
<th>Matrix-size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>50+100</td>
<td>5+5</td>
<td>45+95</td>
<td>4</td>
<td>(2,2)</td>
</tr>
<tr>
<td>Balance</td>
<td>288+288</td>
<td>28+28</td>
<td>260+260</td>
<td>4</td>
<td>(2,2)</td>
</tr>
<tr>
<td>PimaIndians</td>
<td>268+500</td>
<td>25+25</td>
<td>243+475</td>
<td>8</td>
<td>(2,4)</td>
</tr>
<tr>
<td>Heart</td>
<td>120+150</td>
<td>15+15</td>
<td>105+135</td>
<td>13</td>
<td>(3,5)</td>
</tr>
<tr>
<td>Australian</td>
<td>383+307</td>
<td>40+40</td>
<td>343+267</td>
<td>14</td>
<td>(4,4)</td>
</tr>
<tr>
<td>Wine</td>
<td>59+71</td>
<td>7+7</td>
<td>52+64</td>
<td>13</td>
<td>(3,5)</td>
</tr>
<tr>
<td>Lung</td>
<td>10+13</td>
<td>3+3</td>
<td>7+10</td>
<td>56</td>
<td>(7,8)</td>
</tr>
<tr>
<td>Letters</td>
<td>789+766</td>
<td>70+70</td>
<td>719+696</td>
<td>(4,4)</td>
<td>(4,4)</td>
</tr>
<tr>
<td>Robot</td>
<td>20+27</td>
<td>4+4</td>
<td>16+23</td>
<td>(15,6)</td>
<td>(15,6)</td>
</tr>
<tr>
<td>Libras</td>
<td>24+24</td>
<td>4+4</td>
<td>20+20</td>
<td>(45,2)</td>
<td>(45,2)</td>
</tr>
<tr>
<td>Hand</td>
<td>20+20</td>
<td>4+4</td>
<td>16+16</td>
<td>(16,16)</td>
<td>(16,16)</td>
</tr>
<tr>
<td>Eyes</td>
<td>40+40</td>
<td>5+5</td>
<td>35+35</td>
<td>(84,56)</td>
<td>(84,56)</td>
</tr>
</tbody>
</table>
Table 6. Averaged training time and best parameters on various datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>SVM</th>
<th>STM</th>
<th>ν-TSVM</th>
<th>ν-TSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time C</td>
<td>Time C</td>
<td>Time ν</td>
<td>Time ν</td>
</tr>
<tr>
<td>Iris</td>
<td>0.001 2^(-1)</td>
<td>0.843 2^(-1)</td>
<td>0.007 0.1</td>
<td>0.549 0.3</td>
</tr>
<tr>
<td>Balance</td>
<td>0.095 2^(-5)</td>
<td>7.925 2^(-5)</td>
<td>0.027 0.2</td>
<td>0.946 0.6</td>
</tr>
<tr>
<td>PimaIndians</td>
<td>0.101 2^4</td>
<td>4.139 2^3</td>
<td>0.03 0.4</td>
<td>0.87 0.5</td>
</tr>
<tr>
<td>Heart</td>
<td>0.062 2^(-5)</td>
<td>2.277 2^(-1)</td>
<td>0.013 0.3</td>
<td>0.595 0.3</td>
</tr>
<tr>
<td>Australian</td>
<td>0.156 2^5</td>
<td>9.641 2^(-4)</td>
<td>0.052 0.6</td>
<td>1.706 0.3</td>
</tr>
<tr>
<td>Wine</td>
<td>0.001 2^1</td>
<td>0.546 2^0</td>
<td>0.008 0.9</td>
<td>0.524 0.2</td>
</tr>
<tr>
<td>Lung</td>
<td>0.001 2^(-1)</td>
<td>0.515 2^0</td>
<td>0.008 0.3</td>
<td>0.398 0.2</td>
</tr>
<tr>
<td>Letters</td>
<td>0.655 2^(-3)</td>
<td>3.65 2^0</td>
<td>0.139 0.2</td>
<td>3.909 0.2</td>
</tr>
<tr>
<td>Robot</td>
<td>0.001 2^(-1)</td>
<td>0.515 2^5</td>
<td>0.011 0.8</td>
<td>0.494 0.9</td>
</tr>
<tr>
<td>Libras</td>
<td>0.001 2^2</td>
<td>0.65 2^0</td>
<td>0.009 0.3</td>
<td>1.411 0.6</td>
</tr>
<tr>
<td>Hand</td>
<td>0.001 2^(-5)</td>
<td>0.452 2^2</td>
<td>0.027 0.1</td>
<td>0.671 0.9</td>
</tr>
<tr>
<td>Eyes</td>
<td>0.001 2^(-5)</td>
<td>0.624 2^(-5)</td>
<td>75.026 0.1</td>
<td>0.815 0.5</td>
</tr>
</tbody>
</table>

Table 6 and Table 7 show the experiments results. Table 6 records the averaged running time and the best parameters on four algorithms in many times experiments. We have an overall view in these 12 datasets results. Although the proposed ν-TSTM works slower than SVM and ν-TSVM, it works faster than STM mostly. With the dimension increasing, the solving speed of ν-TSTM becomes faster comparatively. It means that the proposed ν-TSTM is more suitable for high dimensional and S3 problem. The results of testing accuracy in Table 7 indicate that ν-TSTM performs best on various datasets, followed by STM, ν-TSVM and SVM. The sensitivity and specificity results show that tensor-based algorithms ν-TSTM and STM perform outstanding than vector-based algorithms ν-TSVM and SVM. Tensor-based algorithms are better than vector-based ones in 15 out of 24 comparisons. And In tensor-based algorithm ν-TSVM takes 11 out of 15 comparisons. The results verify that tensor-based algorithms can overcome the overfitting problem caused mostly by vector-based algorithms.
Table 7. Averaged testing accuracy, sensitivity and specificity on various datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>SVM</th>
<th>STT</th>
<th>ν-TSVM</th>
<th>ν-TSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>Accuracy</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
</tr>
<tr>
<td>Balance</td>
<td>Accuracy</td>
<td>89.13±5.20</td>
<td>83.64±13.85</td>
<td>97.17±1.97</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>92.78±7.13</td>
<td>68.15±28.38</td>
<td>100.00±0.00</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>86.00±9.07</td>
<td>90.08±3.44</td>
<td>94.76±3.40</td>
</tr>
<tr>
<td>PimaIndians</td>
<td>Accuracy</td>
<td>72.22±3.69</td>
<td>70.71±1.71</td>
<td>71.36±2.44</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>37.53±22.09</td>
<td>69.58±13.09</td>
<td>71.98±6.78</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>89.97±8.55</td>
<td>71.29±6.84</td>
<td>71.05±6.43</td>
</tr>
<tr>
<td>Heart</td>
<td>Accuracy</td>
<td>75.91±6.68</td>
<td>78.47±3.18</td>
<td>76.45±2.73</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>79.11±16.33</td>
<td>78.52±6.33</td>
<td>79.25±10.21</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>71.81±16.77</td>
<td>78.41±2.39</td>
<td>72.85±12.15</td>
</tr>
<tr>
<td>Australian</td>
<td>Accuracy</td>
<td>67.37±5.96</td>
<td>71.47±2.41</td>
<td>84.54±2.29</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>72.53±21.84</td>
<td>77.84±1.77</td>
<td>80.87±4.71</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>60.74±29.71</td>
<td>63.29±7.79</td>
<td>89.25±3.20</td>
</tr>
<tr>
<td>Wine</td>
<td>Accuracy</td>
<td>87.41±9.68</td>
<td>92.67±4.25</td>
<td>75.68±8.67</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>76.15±25.21</td>
<td>87.5±12.71</td>
<td>75.38±35.41</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>95.65±5.22</td>
<td>96.87±2.85</td>
<td>75.93±14.34</td>
</tr>
<tr>
<td>Lung</td>
<td>Accuracy</td>
<td>68.23±8.92</td>
<td>69.12±16.89</td>
<td>70.58±4.80</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>62.00±22.8</td>
<td>70.00±40.82</td>
<td>72.50±9.57</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>77.14±16.28</td>
<td>67.85±24.39</td>
<td>50.00±30.3</td>
</tr>
<tr>
<td>Letters</td>
<td>Accuracy</td>
<td>94.36±1.29</td>
<td>96.85±0.45</td>
<td>98.83±0.15</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>88.90±2.55</td>
<td>94.08±0.49</td>
<td>98.88±0.98</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>100.00±0.00</td>
<td>99.71±0.41</td>
<td>98.77±1.32</td>
</tr>
<tr>
<td>Robot</td>
<td>Accuracy</td>
<td>60.51±6.68</td>
<td>62.18±6.05</td>
<td>66.67±2.09</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>93.75±10.82</td>
<td>96.87±6.25</td>
<td>89.06±10.67</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>37.39±12.52</td>
<td>38.04±10.86</td>
<td>51.08±4.16</td>
</tr>
<tr>
<td>Libras</td>
<td>Accuracy</td>
<td>64.50±8.17</td>
<td>64.37±5.15</td>
<td>70.50±6.70</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>50.00±30.82</td>
<td>83.75±16.52</td>
<td>64.00±18.84</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>79.00±26.07</td>
<td>45.00±15.81</td>
<td>77.00±21.96</td>
</tr>
<tr>
<td>Hand</td>
<td>Accuracy</td>
<td>99.38±1.39</td>
<td>99.22±1.56</td>
<td>96.25±2.61</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>98.75±2.79</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>100.00±0.00</td>
<td>98.44±3.125</td>
<td>82.50±5.23</td>
</tr>
<tr>
<td>Eyes</td>
<td>Accuracy</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
</tr>
<tr>
<td></td>
<td>Specificity</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
<td>100.00±0.00</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

In this work, we propose a novel tensor based algorithm named $\nu$-Twin Support Tensor Machine. Compared with vector-based algorithms, $\nu$-TSTM inherits the advantage of STM that utilizes the data structural information. It is crucial and beneficial to classify the category exactly and reduce the number of variables. Similar to $\nu$-TSVM, $\nu$-TSTM solves a pair of non-parallel smaller-sized QPPs to reduce its computational complexity. And the pair of parameters ($\nu$) are used to control the bounds of the fractions of support tensors and the error margins. In addition, $\nu$-TSTM can avoid overfitting problems and more suit for high dimensional and small-sample-size problems. As respected, the computational experiments on vector-based datasets and tensor-based datasets testified the mentioned advantages.

However, there are some drawbacks of the proposed algorithm. Firstly, the processing of solving parameters is still time consuming. Besides, due to the Alternate Iterating Algorithm, the training time of $\nu$-TSTM is much more than vector-based algorithms. For the computing speed, the algorithm still remains to be further optimized. In the future, the possible direction of our work is to investigate more efficient computational methods for solving the optimization problems of $\nu$-TSTM. In addition, an interesting topic is to develop $\nu$-TSTM to nonlinear case and design algorithms to solve high-rank, high-order $\nu$-TSTM.

References


