One-Class Support Tensor Machine

Yanyan Chen^{1,2}, Kuaini Wang³, Ping Zhong^{* 1}

¹ *College of Science, China Agricultural University, Beijing, 100083, China*

² *College of Applied Science and Technology, Beijing Union University, Beijing 102200, China*

³ *College of Science, Xi'an Shiyou University, Xi'an 710065, China*

Abstract In fault diagnosis, face recognition, network anomaly detection, text classification and many other fields, we often encounter one-class classification problems. The traditional vector-based one-class classification algorithms represented by one-class Support Vector Machine (OCSVM) have limitations when tensor is considered as input data. This work addresses one-class classification problem with tensor-based maximal margin classification paradigm. To this end, we formulate the One-Class Support Tensor Machine (OCSTM), which separates most samples of interested class from the origin in the tensor space, with maximal margin. The benefits of the proposed algorithm are twofold. First, the use of direct tensor representation helps to retain the data topology more efficiently. The second benefit is that tensor representation can greatly reduce the number of parameters. It helps overcome the overfitting problem caused mostly by vector-based algorithms and especially suits for high dimensional and small sample size problem. To solve the corresponding optimization problem in OCSTM, the alternating projection method is implemented, for it is simplified by solving a typical OCSVM optimization problem at each iteration. The efficiency of the proposed method is illustrated on both vector and tensor datasets. The experimental results indicate the validity of the new method.

Keywords: Support vector machine; Support tensor machine; One-class classification; High dimensional and small sample size problem.

^{1*}Corresponding author. E-mail: zping@cau.edu.cn(Ping Zhong).

1 Introduction

1.1 Problem Statement

In many practical application fields, we often encounter one-class classification problems, such as fault diagnosis, face recognition, the network anomaly detection and text classification etc. In one-class classification problems, usually, only one class is available, and others are either expensive to acquire or difficult to characterize. Support Vector Machine (SVM) [1] had been extended to one-class classification by Scholkopf et al. [2], termed as One-Class Support Vector Machine (OCSVM). OCSVM seeks for a hyperplane in the maximal margin sense, which separates most of the samples from the origin. Another approach derived from SVM to solve the one-class classification problem is the Support Vector Domain Description (SVDD) [3]. SVDD maps the data into a feature space and consists in determining the minimum volume enclosing ball which contains almost all the samples. At present, there are lots of researches on one-class classification problems [4, 5, 6, 7], and all of which are based on vector space.

In pattern recognition, machine learning, computer vision and image processing and other fields, many of the original objects are represented as multidimensional arrays, namely tensor. For example, gray face image [8] is represented as a second order tensor (or matrix); color image and grayscale video sequence [9], gait contour sequence [10] and hyper spectral cube [11] are usually expressed as third order tensor; color video sequence is usually expressed as fourth order tensor. The traditional one-class classification methods represented by OCSVM failed when tensor is considered as input data. Although there are some methods converting tensor directly into vector, they may lead to structural information loss and data correlation damage [12]. All these reasons lead us to consider tensor representations and corresponding learning algorithms for one-class classification problems.

1.2 Related Work

Based on SVM learning framework and the ideas of alternating projection and multiple linear algebra operations, Tao and Wu processed Supervised Tensor Learning (STL) framework [13], which took tensor as input data. Tensor representation can effectively reduce the overfitting problem in the traditional learning methods of vector representation. Based on STL framework and its bidirectional optimal projection algorithm, Tao and Cai proposed Support Tensor Machine (STM) [14]. Since the parameters to be solved in tensor based classifier are far less than those in vector-based classifier, STM is especially suitable for small sample size problem. The experiment results also show that classification accuracy of STM is superior to that of the traditional SVM.

In recent years, there have been more and more researches on evolving the vector-based learning algorithm to tensor representation, and all have obtained good experimental

results and applications. Cai et al. presented a linear Tensor Least Square(TLS) classifier with second order tensor, and applied STM for text classification [15]. Tao et al. extended the classic linear C-SVM [16] and ν -SVM [17] to the general tensor forms [12]. Zhang et al. generalized the vector-based learning algorithm Twin Support Vector Machine (TWSVM) to the tensor-based method Twin Support Tensor Machine (TWSTM), and implemented the classifier for micro calcification clusters detection [18]. By comparison with TWSVM, the tensor version reduces the overfitting problem significantly. Reshma et al. developed a least squares variant of STM, termed as Proximal Support Tensor Machine (PSTM) [19]. Irene Kotsia et al. formulated the higher rank Support Tensor Machines (STMs) in which the separating hyperplane was constrained to be the sum of rank one tensors [20]. Hao et al. presented a novel linear Support Higher-order Tensor Machine (SHTM) which integrated the merits of linear C-SVM and tensor rank-one decomposition [21]. SHTM solved the problem of time-consuming caused by STM and was particularly effective for higher-order tensor. On the similar line, Liu et al. extended the concave-convex procedure-based Transductive Support Vector Machine (CCCP-TSVM) to tensor pattern CCCP-based Transductive Support Tensor Machine (CCCP-TSTM), in which the tensor rank-one decomposition was used to compute the inner product of the tensors [22].

As to the nonlinear cases, there are several studies on kernel methods for tensors. Signoretto et al. elaborated on a possible framework to extend the flexibility of tensor-based models with kernel-based techniques, which gave a constructive definition to the feature space of infinite dimensional tensors [23]. He et al. introduced a new scheme to design structure-preserving kernels for supervised tensor learning [24]. For specified order of tensors, there are some even simple and convenient calculation kernel methods, i.e. matrix kernel function for second order tensor [25, 26] and K_{3rd} kernel function for third order tensor [27].

All the relevant literatures show that the tensor representation of the data can remain the natural structure and correlation of the data, and avoid information missing, overfitting and curse of dimensionality. Therefore, the studies of the tensor have attracted more and more attentions, and the researchers have extended the framework of the support vector machine to the tensor patterns and have proposed many tensor-based learning machines.

1.3 Research Contribution

Utilizing the advantages of STL framework and one-class Support Vector Machine, we derive a tensor-based one-class classification algorithm, named One-Class Support Tensor Machine (OCSTM). The main idea of OCSTM is to find a hyperplane in tensor space (or tensor feature space), which can separate most samples of the interest class from the origin with the maximal margin principle. Since the matrix kernel function is relatively simple,

we first present our OCSTM with second order tensor. Then we generalize a framework of OCSTM for high order tensors. Based on STL framework and its bidirectional optimal projection algorithm, the corresponding optimization problem in OCSTM is solved in an iterative manner, where at each iteration the parameters corresponding to the projections are estimated by solving a typical OCSVM optimization problem. This new model takes tensor as input data, so that it can utilize the structural information which is presented in multidimensional features of an object. Since tensor representation can greatly reduce the number of parameters estimated by OCSVM, our OCSTM helps overcome the overfitting problem encountered mostly in vector-based algorithms and especially suits for high dimensional and small sample size problem.

To evaluate the new algorithm, two kinds of datasets are considered: vector-based and tensor-based datasets. For vector-based datasets, since the vector data points need to be converted to tensor form, we first discuss on how to choose the proper tensor size. Then we test the classification performance of our OCSTM by comparison with OCSVM. Since tensor representation is particular suitable for high dimensional and small sample size cases, we detail the performance of each classifier with different small sample sizes. We also discuss on time cost and overfitting problem on vector-based dataset. The considered tensor-based dataset is human face image, which can be represented as a second order tensor and always be joined lines into vector in traditional vector learning algorithms. The experiments in this part indicate that tensor representation can reserve the important structure attributes of the image, which vector-based methods cannot qualify. All the experimental results indicate the validity and advantage of the new OCSTM.

The rest of this paper is organized as follows: In Section 2, we give a brief overview of Support Tensor Machine and the kernel function for tensor. The One-Class Support Tensor Machine, is described in Section 3. The experimental results on vector datasets and tensor datasets are presented in Section 4 and Section 5. In Section 6, we generalize a framework of OCSTM for high order tensors. Finally, we provide some concluding remarks and suggestions for future work in Section 7.

2 Brief Overview of Support Tensor Machine

2.1 Support Tensor Machine

Support Tensor Machine, which was developed by Cai and Tao [13, 14], is a tensor generalization of SVM in the tensor space. Suppose the training samples $\{\mathbf{X}_i, y_i\} (i = 1, \dots, l)$, where $\mathbf{X}_i \in \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2}$ is the data point in 2nd-order tensor space, $y_i \in \{-1, 1\}$ is the class associated with $\mathbf{X}_i$, $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$ are two vector spaces. STM tries to find the following linear classifier in the tensor space:

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

such that the two classes can be separated with maximal margin.

The optimization problem of linear STM can be stated as:

$$\begin{aligned}
\min_{\mathbf{u} \in \mathbb{R}^{n_1}, \mathbf{v} \in \mathbb{R}^{n_2}, b \in \mathbb{R}, \xi \in \mathbb{R}^l} \quad & \frac{1}{2} \|\mathbf{u}\mathbf{v}^T\|^2 + C \sum_{i=1}^l \xi_i \\
\text{s.t.} \quad & y_i(\mathbf{u}^T \mathbf{X}_i \mathbf{v} + b) \geq 1 - \xi_i \\
& \xi_i \geq 0, i = 1, \dots, l
\end{aligned} \tag{2}$$

In order to solve the optimization problem (2), Cai and Tao describe a simple yet effective computational method. To fix $\mathbf{u}$ at first, let $\beta_1 = \|\mathbf{u}\|^2$ and $\mathbf{x}_i = \mathbf{X}_i^T \mathbf{u}$. Thus, the optimization problem (2) is identical to the standard SVM optimization problem with variable $\mathbf{v}$:

$$\begin{aligned}
\min_{\mathbf{v} \in \mathbb{R}^{n_2}, b \in \mathbb{R}, \xi \in \mathbb{R}^l} \quad & \frac{1}{2} \beta_1 \|\mathbf{v}\|^2 + C \sum_{i=1}^l \xi_i \\
\text{s.t.} \quad & y_i(\mathbf{v}^T \mathbf{x}_i + b) \geq 1 - \xi_i \\
& \xi_i \geq 0, i = 1, \dots, l
\end{aligned} \tag{3}$$

It is clear that the optimization problem (3) can be solved by using the same computational methods of SVM. While $\mathbf{v}$ is obtained, let $\beta_2 = \|\mathbf{v}\|^2$ and $\tilde{\mathbf{x}}_i = \mathbf{X}_i \mathbf{v}$, optimization problem (2) is the standard SVM with variable $\mathbf{u}$ as well:

$$\begin{aligned}
\min_{\mathbf{u} \in \mathbb{R}^{n_1}, b \in \mathbb{R}, \xi \in \mathbb{R}^l} \quad & \frac{1}{2} \beta_2 \|\mathbf{u}\|^2 + C \sum_{i=1}^l \xi_i \\
\text{s.t.} \quad & y_i(\mathbf{u}^T \tilde{\mathbf{x}}_i + b) \geq 1 - \xi_i \\
& \xi_i \geq 0, i = 1, \dots, l
\end{aligned} \tag{4}$$

Then $\mathbf{u}$ and $\mathbf{v}$ can be obtained by iteratively solving the standard SVM optimization problems (3) and (4).

2.2 The Kernel Function for Tensor

Since STM is a linear classifier in the tensor space, it cannot handle nonlinear data. Gao proposed a kernel function for tensor representation data [25], which uses a nonlinear mapping function $\Phi(\mathbf{X}_i)$ to map $\mathbf{X}_i$ into a high dimensional tensor feature space. The nonlinear mapping function for tensor $\mathbf{X}_i$ can be defined as:

$$\Phi(\mathbf{X}_i) = \begin{bmatrix} \varphi(z_{i1}) \\ \varphi(z_{i2}) \\ \vdots \\ \varphi(z_{in_1}) \end{bmatrix}, \tag{5}$$

where z_{ip} is the p -th row of $\mathbf{X}_i$. Thus the new kernel function for tensors can be described as:

$$\begin{aligned} K(\mathbf{X}_i, \mathbf{X}_j) &= \Phi(\mathbf{X}_i)\Phi(\mathbf{X}_j)^T = \begin{bmatrix} \varphi(z_{i1}) \\ \varphi(z_{i2}) \\ \vdots \\ \varphi(z_{in_1}) \end{bmatrix} \begin{bmatrix} \varphi(z_{j1}) \\ \varphi(z_{j2}) \\ \vdots \\ \varphi(z_{jn_1}) \end{bmatrix}^T \\ &= \begin{bmatrix} \varphi(z_{i1})\varphi(z_{j1})^T & \dots & \varphi(z_{i1})\varphi(z_{jn_1})^T \\ \vdots & \ddots & \vdots \\ \varphi(z_{in_1})\varphi(z_{j1})^T & \dots & \varphi(z_{in_1})\varphi(z_{jn_1})^T \end{bmatrix}. \end{aligned} \quad (6)$$

The kernel function for tensors is different from the one for vectors. In vector space, the result of kernel function is a scalar and in tensor space it is a kernel matrix. Specifically, we use RBF kernel function in this article, since it has been demonstrated that the RBF kernel usually outperforms than other kernels [28]. The ij -th element of the kernel matrix is:

$$\varphi(z_{ip_1})\varphi(z_{jp_2})^T = e^{-\|z_{ip_1} - z_{jp_2}\|^2/2\sigma^2}, \quad (7)$$

we call it the RBF kernel function for Tensors, shorted as TRBF kernel matrix in the following passage.

3 One-Class Support Tensor Machine

One-Class Support Vector Machine aims to learn a single class by determining a decision function with maximal margin from the origin that contains almost all the data of this class. Usually, we call the available class as the target class, while all other instances not in this class are defined as outliers. Consider training data $\mathbf{x}_i \in \mathbb{R}^n (i = 1, \dots, l)$, the decision function relative to the membership of the sample $\mathbf{x}$ to the target class is given by: $f(\mathbf{x}) = ((\mathbf{w} \cdot \mathbf{x}) - \rho) \geq 0$, where parameters $\mathbf{w}$ and ρ result from the modified optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, \xi, \rho} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu l} \sum_{i=1}^l \xi_i - \rho \\ \text{s.t.} \quad & (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i \\ & \xi_i \geq 0, i = 1, \dots, l \end{aligned} \quad (8)$$

Here, the favorable parameter ν can control the fraction of support vectors, and ξ_i are slack variables which allow discarding outliers.

Our One-Class Support Tensor Machine is fundamentally based on the same idea. In this section, we propose a new one-class classifier OCSTM based on tensor space, which determines a decision function by mapping the tensor data into high dimensional tensor

feature space, so that the data points in the target class are separated by maximal margin from the origin.

3.1 One-Class Support Tensor Machine

Suppose we are given a set of training samples $\{\mathbf{X}_i\}(i = 1, \dots, l)$, each of the training sample $\mathbf{X}_i \in \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2}$ is the data point in 2nd-order tensor space, where $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$ are two vector spaces.

As similar as OCSVM, the decision function of an one-class classifier in the tensor space can be represented as follows:

$$f(\mathbf{X}) = \text{sgn}(\mathbf{u}^T \Phi(\mathbf{X}) \mathbf{v} - \rho), \mathbf{u} \in \mathbb{R}^{n_1}, \mathbf{v} \in \mathbb{R}^{n_2}, \quad (9)$$

where Φ is a function mapping the data from the original tensor space to a tensor feature space.

Equation (9) can be rewritten through matrix inner product as follows:

$$f(\mathbf{X}) = \text{sgn}((\mathbf{u} \mathbf{v}^T \cdot \Phi(\mathbf{X})) - \rho), \mathbf{u} \in \mathbb{R}^{n_1}, \mathbf{v} \in \mathbb{R}^{n_2}. \quad (10)$$

Our One-Class Support Tensor Machine can be given by the following optimization problem:

$$\begin{aligned} \min_{\mathbf{u} \in \mathbb{R}^{n_1}, \mathbf{v} \in \mathbb{R}^{n_2}, \rho \in \mathbb{R}, \xi \in \mathbb{R}^l} \quad & \frac{1}{2} \|\mathbf{u} \mathbf{v}^T\|^2 + \frac{1}{\nu l} \sum_{i=1}^l \xi_i - \rho \\ \text{s.t.} \quad & (\mathbf{u} \mathbf{v}^T \cdot \Phi(\mathbf{X}_i)) \geq \rho - \xi_i \\ & \xi_i \geq 0, i = 1, \dots, l \end{aligned} \quad (11)$$

We introduce positive Lagrange multipliers $\alpha_i, \beta_i \geq 0, i = 1, \dots, l$, one for each of the inequality constrains. The Lagrangian function for problem (11) is

$$\begin{aligned} \mathcal{L}(\mathbf{u}, \mathbf{v}, \rho, \xi, \alpha, \beta) = & \frac{1}{2} \|\mathbf{u} \mathbf{v}^T\|^2 + \frac{1}{\nu l} \sum_{i=1}^l \xi_i - \rho \\ & - \sum_{i=1}^l \alpha_i ((\mathbf{u} \mathbf{v}^T \cdot \Phi(\mathbf{X}_i)) - \rho + \xi_i) - \sum_{i=1}^l \beta_i \xi_i. \end{aligned} \quad (12)$$

Note that:

$$\begin{aligned} \frac{1}{2} \|\mathbf{u} \mathbf{v}^T\|^2 &= \frac{1}{2} \text{trace}(\mathbf{u} \mathbf{v}^T \mathbf{v} \mathbf{u}^T) \\ &= \frac{1}{2} (\mathbf{v}^T \mathbf{v}) \text{trace}(\mathbf{u} \mathbf{u}^T) \\ &= \frac{1}{2} (\mathbf{v}^T \mathbf{v}) (\mathbf{u}^T \mathbf{u}). \end{aligned} \quad (13)$$

Thus, we have

$$\begin{aligned}\mathcal{L}(\mathbf{u}, \mathbf{v}, \rho, \xi, \alpha, \beta) &= \frac{1}{2}(\mathbf{v}^T \mathbf{v})(\mathbf{u}^T \mathbf{u}) + \frac{1}{\nu l} \sum_{i=1}^l \xi_i - \rho \\ &\quad - \sum_{i=1}^l \alpha_i (\mathbf{u}^T \Phi(\mathbf{X}_i) \mathbf{v} - \rho + \xi_i) - \sum_{i=1}^l \beta_i \xi_i.\end{aligned}\quad (14)$$

Setting the derivatives with respect to the primal variables $\mathbf{u}, \mathbf{v}, \xi_i, \rho$ equal to zero, we have

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = 0 \Rightarrow \mathbf{u} = \frac{1}{\|\mathbf{v}\|^2} \sum_{i=1}^l \alpha_i \Phi(\mathbf{X}_i) \mathbf{v} \quad (15)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = 0 \Rightarrow \mathbf{v} = \frac{1}{\|\mathbf{u}\|^2} \sum_{i=1}^l \alpha_i \Phi(\mathbf{X}_i)^T \mathbf{u} \quad (16)$$

$$\frac{\partial \mathcal{L}}{\partial \rho} = 0 \Rightarrow \sum_{i=1}^l \alpha_i = 1 \quad (17)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow \frac{1}{\nu l} - \alpha_i - \beta_i = 0 \quad (18)$$

From Equations (15) and (16), we see that $\mathbf{u}$ and $\mathbf{v}$ are dependent on each other, and can not be solved independently. Hence, we resort to the alternating projection method for solving this optimization problem, which has earlier been used by Cai [14] in developing STM. The method can be described as follows.

First we fix $\mathbf{u}$. Let $\mu_1 = \|\mathbf{u}\|^2$ and $\mathbf{x}_j = \Phi(\mathbf{X}_j)^T \mathbf{u}$, according to (11), we can construct the optimal quadratic programming problem to solve $\mathbf{v}$ and $\|\mathbf{v}\|^2$:

$$\begin{aligned}\min_{\mathbf{v}, \xi, \rho} \quad & \frac{1}{2} \mu_1 \|\mathbf{v}\|^2 + \frac{1}{\nu l} \sum_{j=1}^l \xi_j - \rho \\ \text{s.t.} \quad & (\mathbf{v} \cdot \mathbf{x}_j) \geq \rho - \xi_j \\ & \xi_j \geq 0, j = 1, \dots, l\end{aligned}\quad (19)$$

It can be seen that the optimization problem (19) is similar in structure to OCSVM. For solving (19), we consider its Lagrangian function

$$\begin{aligned}\mathcal{L}(\mathbf{v}, \rho, \xi, \alpha, \beta) &= \frac{1}{2} \mu_1 (\mathbf{v}^T \mathbf{v}) + \frac{1}{\nu l} \sum_{j=1}^l \xi_j - \rho \\ &\quad - \sum_{j=1}^l \alpha_j (\mathbf{v}^T \mathbf{x}_j - \rho + \xi_j) - \sum_{j=1}^l \beta_j \xi_j.\end{aligned}\quad (20)$$

According to Equations (15) to (18),

$$\begin{aligned}
\mathcal{L}(\mathbf{v}, \rho, \xi, \alpha, \beta) &= \frac{1}{2}\mu_1(\mathbf{v}^T \mathbf{v}) - \sum_{j=1}^l \alpha_j \mathbf{v}^T \mathbf{x}_j \\
&= -\frac{1}{2\mu_1} \sum_{i,j=1}^l \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\
&= -\frac{1}{2\mu_1} \sum_{i,j=1}^l \alpha_i \alpha_j (\Phi(\mathbf{X}_i)^T \mathbf{u} \cdot \Phi(\mathbf{X}_j)^T \mathbf{u}) \\
&= -\frac{1}{2\mu_1} \sum_{i,j=1}^l \alpha_i \alpha_j \mathbf{u}^T K(\mathbf{X}_i, \mathbf{X}_j) \mathbf{u},
\end{aligned} \tag{21}$$

where $K(\mathbf{X}_i, \mathbf{X}_j)$ is defined by Equation(6).

Thus we can get the dual problem of optimization problem (19):

$$\begin{aligned}
\min_{\alpha} \quad & \frac{1}{2\mu_1} \sum_{i,j=1}^l \alpha_i \alpha_j \mathbf{u}^T K(\mathbf{X}_i, \mathbf{X}_j) \mathbf{u} \\
\text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu l} \\
& \sum_{i=1}^l \alpha_i = 1, i = 1, \dots, l
\end{aligned} \tag{22}$$

Solving (22) determines the lagrangian multipliers α_i^* , then we can get $\mathbf{v}$ and $\|\mathbf{v}\|^2$:

$$\mathbf{v} = \frac{1}{\mu_1} \sum_{i=1}^l \alpha_i^* \Phi(\mathbf{X}_i)^T \mathbf{u} \tag{23}$$

$$\|\mathbf{v}\|^2 = \mathbf{v}^T \mathbf{v} = \frac{1}{\mu_1^2} \sum_{i,j=1}^l \alpha_i^* \alpha_j^* \mathbf{u}^T K(\mathbf{X}_i, \mathbf{X}_j) \mathbf{u} \tag{24}$$

Then we can calculate $\mathbf{u}$, according to the result of the (23) and (24). We can let $\mu_2 = \|\mathbf{v}\|^2$ and

$$\begin{aligned}
\mathbf{x}_j' = \Phi(\mathbf{X}_j) \mathbf{v} &= \frac{1}{\mu_1} \sum_{i=1}^l \alpha_i^* \Phi(\mathbf{X}_j) \Phi(\mathbf{X}_i)^T \mathbf{u} \\
&= \frac{1}{\mu_1} \sum_{i=1}^l \alpha_i^* K(\mathbf{X}_j, \mathbf{X}_i) \mathbf{u}
\end{aligned} \tag{25}$$

On the similar lines, we can construct another optimal quadratic programming problem

to solve $\mathbf{u}$ and $\|\mathbf{u}\|^2$:

$$\begin{aligned} \min_{\mathbf{u}, \xi, \rho} \quad & \frac{1}{2}\mu_2\|\mathbf{u}\|^2 + \frac{1}{\nu l} \sum_{j=1}^l \xi_j - \rho \\ \text{s.t.} \quad & (\mathbf{u} \cdot \mathbf{x}_j') \geq \rho - \xi_j \\ & \xi_j \geq 0, j = 1, \dots, l \end{aligned} \quad (26)$$

For solving (26), the quadratic program is solved via its dual:

$$\begin{aligned} \min_{\hat{\alpha}} \quad & \frac{1}{2\mu_2} \sum_{i,j=1}^l \hat{\alpha}_i \hat{\alpha}_j \mathbf{x}_i'^T \mathbf{x}_j' \\ \text{s.t.} \quad & 0 \leq \hat{\alpha}_i \leq \frac{1}{\nu l} \\ & \sum_{i=1}^l \hat{\alpha}_i = 1, i = 1, \dots, l \end{aligned} \quad (27)$$

Thus, $\mathbf{u}$ and $\mathbf{v}$ can be obtained by iteratively solving the optimization problems (19) and (26). In our experiments, $\mathbf{u}$ is initially set to the vector of all ones.

The optimal normal vector from optimization problem (26) is given by $\mathbf{u} = \frac{1}{\mu_2} \sum_{i=1}^l \hat{\alpha}_i^* \mathbf{x}_i'$, where $\hat{\alpha}_i^*$ is the result of dual problem (27). The optimal boundary is then determined by the support vector expansion:

$$\begin{aligned} f(\mathbf{X}) &= \text{sgn}((\mathbf{u}\mathbf{v}^T \cdot \Phi(\mathbf{X})) - \rho) \\ &= \text{sgn}\left(\frac{1}{\mu_1} \sum_{i=1}^l \alpha_i^* \mathbf{u}^T K(\mathbf{X}, \mathbf{X}_i) \mathbf{u} - \rho\right), \end{aligned} \quad (28)$$

where training samples $\mathbf{X}_i$ with non-zero α_i^* are support tensors, which are labeled as $\mathbf{X}_{i_sv}$. And parameters ρ is calculated by:

$$\rho = \text{mean}_{i_sv} \left(\frac{1}{\mu_1} \sum_{i=1}^l \alpha_i^* \mathbf{u}^T K(\mathbf{X}_i, \mathbf{X}_{i_sv}) \mathbf{u} \right). \quad (29)$$

3.2 The Algorithm of OCSTM

OCSTM is a tensor generalization of OCSVM. The steps involved in solving OCSTM use the alternating projection method, and the algorithmic procedure is formally stated below:

Input: The training samples $\mathbf{X}_i \in \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2} (i = 1, \dots, l)$ and testing samples $\mathbf{X}_j \in \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2} (j = 1, \dots, t)$, the parameters ν and σ .

Output: The optimal parameters $\mathbf{u} \in \mathbb{R}^{n_1}$ and ρ in discrimination function, and the class labels of testing samples.

Step 1 Initialization: Let $\mathbf{u} = (1, \dots, 1)^T$;

Step 2 Obtain α by optimizing

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2\mu_1} \sum_{i,j=1}^l \alpha_i \alpha_j \mathbf{u}^T K(\mathbf{X}_i, \mathbf{X}_j) \mathbf{u} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu l} \\ & \sum_{i=1}^l \alpha_i = 1, i = 1, \dots, l \end{aligned} \tag{30}$$

where $\mu_1 = \|\mathbf{u}\|^2$;

Step 3 Calculate μ_2 and $\mathbf{x}_j'$, let:

$$\begin{aligned} \mu_2 &= \frac{1}{\mu_1^2} \sum_{i,j=1}^l \alpha_i^* \alpha_j^* \mathbf{u}^T K(\mathbf{X}_i, \mathbf{X}_j) \mathbf{u}; \\ \mathbf{x}_j' &= \frac{1}{\mu_1} \sum_{i=1}^l \alpha_i^* K(\mathbf{X}_j, \mathbf{X}_i) \mathbf{u}; \end{aligned}$$

Step 4 Solving

$$\begin{aligned} \min_{\hat{\alpha}} \quad & \frac{1}{2\mu_2} \sum_{i,j=1}^l \hat{\alpha}_i \hat{\alpha}_j \mathbf{x}_i'^T \mathbf{x}_j' \\ \text{s.t.} \quad & 0 \leq \hat{\alpha}_i \leq \frac{1}{\nu l} \\ & \sum_{i=1}^l \hat{\alpha}_i = 1, i = 1, \dots, l \end{aligned} \tag{31}$$

obtain the new $\mathbf{u} = \frac{1}{\mu_2} \sum_{i=1}^l \hat{\alpha}_i^* \mathbf{x}_i'$;

Step 5 Do Step2~4 iteratively until convergence: If the iteration number exceeds the maximum number of iteration or the below convergence condition is satisfied:

$$\|\mathbf{u}_i - \mathbf{u}_{i-1}\| \leq \textit{tolerance}; \tag{32}$$

Step 6 Obtain the class label of the testing samples by using the discrimination function

$$f(\mathbf{X}_j) = \textit{sgn}\left(\frac{1}{\mu_1} \sum_{i=1}^l \alpha_i^* \mathbf{u}^T K(\mathbf{X}_j, \mathbf{X}_i) \mathbf{u} - \rho\right), \tag{33}$$

where ρ can be calculated from (29);

Step 7 End.

Now we discuss the computational complexity of OCSTM. Let l be the number of training samples and n be the number of the features, and in OCSVM n also denotes the dimension of vector representation data points. In the case of considering the dimensionality of the data, the computational complexity of OCSVM is $O(l^2n)$, while the computational complexity of OCSTM is $O(l^2(n_1 + n_2))$ with linear kernel and $O(l^2n_1^2n_2)$ with TRBF kernel matrix, where n_1 and n_2 denote the size of second order tensor and $n_1 \times n_2 \approx n$. It is clear that the computational complexity of OCSTM with linear kernel is less than that of OCSVM. Besides, the training time of OCSTM which involves the alternating projection method, depends on the number of iterations.

4 Experimental evaluations on vector-based datasets

In this section, we compare the performance of OCSTM with that of the standard OCSVM on vector-based datasets. After the statements about the datasets and preparation of experiments, we first evaluate the proposed algorithms on BREAST-CANCER dataset for a trail of detailed discussion on classification performance and the problems of time cost and overfitting. Then we give an overall comparison on all vector-based datasets.

4.1 Datasets and Preparation of Experiments

To evaluate the performance of tensor-based classifiers with vector datasets, we perform experiments on eight datasets from publicly available datasets UCI repository download from LIBSVM dataset webpage [29] and one-class classification datasets on David Tax's homepage [30]. These eight datasets are shown in Table 1, where m is the total number of examples, n is the number of features and $n_1 \times n_2$ indicates the tensor size of each dataset. The features in all datasets are scaled to $[-1, 1]$. In one-class classification issues, we only focus on the target class. Table 1 presents the 12 considered target classes and their number of samples.

In all our simulations, we use RBF kernel $k(\mathbf{x}, \mathbf{y}) = \mathbf{e}^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$ in the standard OCSVM, for it has been demonstrated that the RBF kernel usually outperforms than other kernels [28]. For the same reason, we use tensor kernel function $K(\mathbf{X}, \mathbf{Y})$ (TRBF kernel matrix) for OCSTM, which was defined in section 2. Because linear kernel performs well in some cases, we also evaluate the performance of OCSVM with linear kernel $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$ and OCSTM with linear kernel matrix $K(\mathbf{X}, \mathbf{Y}) = \mathbf{X}\mathbf{Y}^T$. To distinguish with nonlinear classifiers, we name the two linear classifiers as LOCSVM and LOCSTM respectively.

We use k -fold cross-validation on the training set to find the best parameters, while the values of k equal to the number of training samples since we focus on small sample size problem. There are two tuning parameters: ν, σ . The possible choices for parameters are $\nu = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and $\sigma = \{2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 2^3, 2^4, 2^5\}$.

Table 1. Information of the vector-based datasets

Dataset	m	n	$n_1 \times n_2$	Source	Target Class	Target Samples
BREAST-CANCER	683	11	3×4	UCI	2	444
					4	239
					1	50
IRIS	150	4	2×2	UCI	2	50
					3	50
IMPORT	159	25	5×5	OC	1	88
IONOSPHERE	351	34	6×6	OC	1	225
LUNG	32	56	7×8	UCI	2	13
					3	10
SONAR	203	61	8×8	OC	1	97
DELFTPUMP AR	720	160	13×13	OC	2	189
USPS	7291	256	16×16	UCI	2	1005

All the algorithms have been implemented in MATLAB R2011b on Windows 7 running on a PC with system configuration Intel Core i3 (2.4 GHz) and 6 GB of RAM.

Many algorithms use test accuracy to evaluate the performance of the classifiers. The AUC, the area under the ROC curve, is always used to measure the performance of a one-class classifier [31]. In our experiments, we consider both test accuracy and AUC as the performance metrics for comparisons.

4.2 Experiments on BREAST-CANCER dataset

There are two target classes which are labeled as 2 or 4 in BREAST-CANCER dataset. We call them target class 1 and target class 2 in the following passage. The numbers of samples in each class are shown in Table 1. In this section, we experiment on each target class of BREAST-CANCER dataset with the four classifiers: OCSTM, LOCSTM, OCSVM and LOCSVM.

It is well known that it is hard to solve the classification problems with small sample size and high dimensions. To verify the effectiveness of the four algorithms, we choose small sizes of training sets in our experiments, similar to the idea of [14, 19]. For statistical significance, the results are averaged over 50 random splits from target class and the average performance are reported. The sizes of the training sets are 2, 4, 6, 8, and 10, respectively. The testing sets consist of the rest samples in the target class and all the outliers. For example, when the size of training set is 2, we randomly choose 2 samples from the target class as training samples, and combine the remaining samples in the target class and all the outliers to form the testing set.

Figure 1: Four possible types of tensor size in BREAST-CANCER dataset.

Table 2. Averaged percentages of AUC in different tensor sizes.

Training size	Algorithms	Tensor size			
		2×5	3×4	4×3	5×2
2	OCSTM	79.76	84.99	84.62	82.61
	LOCSTM	73.40	77.55	74.43	75.45
4	OCSTM	84.78	92.13	89.77	89.36
	LOCSTM	64.36	70.19	64.65	57.64
6	OCSTM	91.16	92.96	92.95	90.96
	LOCSTM	63.61	71.55	70.22	60.21
8	OCSTM	92.18	92.21	94.22	91.96
	LOCSTM	61.86	75.93	78.45	66.08

4.2.1 Experiments on Different Tensor Sizes

In this subsection we discuss on how to choose the size of the tensor. For a vector space sample $\mathbf{x} \in \mathbb{R}^n$, it can be converted to the second order tensor (or, matrix) $\mathbf{X} \in \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2}$, where $n_1 \times n_2 \approx n$. Cai [14] provided a method to establish n_1 and n_2 , which was simplified as minimizing $n_1 + n_2$ while $(n_1 - 1) \times n_2 \leq n \leq n_1 \times n_2$. With such a requirement, there are still many choices of n_1 and n_2 . Figure 1 shows the four possible tensor sizes which are converted from vector data in BREAST-CANCER dataset. Generally all these types can be used. Therefore, it is worth finding out which one is the best.

Our experiments focus on the performance of two tensor-based classifiers OCSTM and LOCSTM with different tensor sizes. Table 2 summarized the averaged AUC of 50 simulations, along with different tensor sizes and different training sizes. We bold the best averaged AUC in Table 2, as can be seen, when training size is 3×4 , both of the two tensor classifiers obtain outstanding performance.

These experiments suggest that, n_1 and n_2 should be as close as possible. In the meanwhile, $n_1 \leq n_2$ can be a good choice and it is particularly suitable for small sample size problem. With this principle, we establish the tensor sizes of all the vector datasets involved in this paper, as is shown in Table 1.

Table 3. Averaged percentages of test accuracy and AUC on various training sample sizes of BREAST-CANCER dataset.

Num	Target Class	metrics	OCSTM	LOCSTM	OCSVM	LOCSVM
2	class 1	accuracy	63.64±15.08	73.74±14.21	43.92±9.82	68.43±13.42
		AUC	99.32±0.16	99.51±0.03	99.48±0.13	99.48±0.05
	class 2	accuracy	69.40±5.17	65.85±16.75	65.20±0.00	65.83±8.75
		AUC	84.62±23.68	77.55±30.05	80.93±28.50	76.64±31.88
4	class 1	accuracy	75.67±13.02	84.16±10.49	59.80±13.51	79.58±10.81
		AUC	98.52±1.92	99.48±0.11	99.43±0.29	99.46±0.06
	class 2	accuracy	78.57±7.83	64.80±24.57	70.59±4.89	63.92±18.67
		AUC	92.13±10.49	70.19±30.92	89.59±15.31	71.35±31.82
6	class 1	accuracy	82.47±10.39	87.03±10.61	71.68±13.16	84.47±9.38
		AUC	98.31±1.84	99.02±3.41	99.16±1.10	99.27±0.98
	class 2	accuracy	83.88±5.95	65.88±26.02	78.16±5.51	68.13±18.7
		AUC	92.96±9.81	71.55±29.03	93.82±6.84	75.58±26.79
8	class 1	accuracy	83.26±10.26	89.02±8.17	76.45±11.65	86.00±8.64
		AUC	98.50±1.82	99.47±0.12	99.33±0.73	99.30±0.62
	class 2	accuracy	84.96±6.65	70.89±23.37	80.90±5.73	70.34±21.13
		AUC	92.21±10.69	75.93±25.09	93.81±7.86	75.91±26.61
10	class 1	accuracy	84.34±10.39	90.62±7.10	79.81±12.11	88.41±7.11
		AUC	98.40±1.84	99.47±0.17	99.22±0.81	99.27±0.65
	class 2	accuracy	87.11±5.18	74.46±21.85	84.21±4.48	75.11±20.05
		AUC	93.22±9.18	79.55±23.37	94.70±6.38	81.67±22.43

4.2.2 Experiments on Classification Performance

In this subsection, we evaluate the classification performance of the four classifiers on BREAST-CANCER dataset with various training sample sizes. Table 3 has summarized the results, including averaged percentages of test accuracy and AUCs of 50 simulations. We can see that linear tensor classifier LOCSTM is outstanding in target class 1 on almost entirely different training sizes with respect to both test accuracy and AUC metrics. In addition, the overall results show that the linear classifiers are better than nonlinear classifiers in target class 1.

On the opposite, experiments on target class 2 of BREAST-CANCER dataset indicate that the nonlinear classifiers are much better than linear classifiers. Besides, we can see in Table 3 that OCSTM has a significant performance with test accuracy on target class 2, and the AUCs of OCSTM are comparable to those of OCSVM. Especially when training set is small, OCSTM has a remarkable performance in comparison to other classifiers. Notice that the standard deviation of OCSVM is zero when training sample number is 2. In all the 50 simulations, the OCSVM classifies all the testing samples into outliers, which means in this situation, OCSVM is failure to identify the target class.

Table 4. Averaged percentages of training time and the number of iterations on various training sample sizes of BREAST-CANCER dataset.

Num	Algorithm	Target Class 1		Target Class 2	
		Training Time(s)	Num of Iterations	Training Time(s)	Num of Iterations
2	OCSTM	0.0511±0.0791	10.0400±9.3459	0.0439±0.0573	8.6400±4.1293
	LOCSTM	0.0194±0.0038	4.8400±0.7656	0.0312±0.0175	7.8000±4.2714
4	OCSTM	0.1016±0.0907	14.3400±12.4222	0.0341±0.0236	5.4000±3.7143
	LOCSTM	0.0327±0.0122	5.2200±0.8154	0.0426±0.0324	6.8600±5.3031
6	OCSTM	0.1287±0.1414	12.6800±14.0225	0.0311±0.0147	3.3000±1.6194
	LOCSTM	0.0458±0.0082	5.4400±0.9510	0.0438±0.0282	5.0600±3.1129
8	OCSTM	0.1146±0.1003	8.6200±7.7219	0.0418±0.0154	3.2400±1.2216
	LOCSTM	0.0623±0.0115	5.4600±0.9733	0.0697±0.0522	5.6000±4.1157
10	OCSTM	0.1336±0.1311	7.0000±6.0136	0.0592±0.0248	3.3200±1.3768
	LOCSTM	0.0996±0.1039	6.5000±6.4373	0.0817±0.0412	4.9200±2.3976

4.2.3 Experiments on Training Time with Different Kernels

In this subsection, we evaluate the time cost of the proposed OCSTM with different kernel matrix. We use OCSTM to denote the proposed algorithm with TRBF kernel matrix and LOCSTM to indicate the one with linear kernel matrix. Table 4 summarized the averaged training time and the corresponding averaged number of iterations over 50 times simulations, with their standard deviations.

As is discussed in Section 3.2, the computational complexity of OCSTM is more than that of LOCSTM. However, as is seen in Table 4, the averaged training time of OCSTM is less than that of LOCSTM when training sets are 4, 6, 8 and 10 on target class 2 of BREAST-CANCER dataset. And the corresponding averaged number of iterations with OCSTM are less than those with LOCSTM as well. Thus, we can conclude that the training time of the proposed algorithm is more related to the number of iterations.

4.2.4 Experiments on Overfitting Problem

In this subsection, we discuss the overfitting problem which is encountered by the vector-based algorithm with high dimensional and small sample size problem. We compare the proposed tensor-based classifier OCSTM with vector-based classifier OCSVM on BREAST-CANCER dataset. Similar to the idea of [32, 33], to verify the effectiveness dealt with overfitting problem, we use these two evaluation indexes: sensitivity and specificity, where:

$$Sensitivity = \frac{NumberofTruePositiveSamples}{NumberofPositiveSamples} \quad (34)$$

$$Specificity = \frac{NumberofTrueNegativeSamples}{NumberofNegativeSamples} \quad (35)$$

Table 5. Averaged percentages of Sensitivity and Specificity on various training sample sizes of BREAST-CANCER dataset.

Num	Algorithm	Target Class 1		Target Class 2	
		Sensitivity	Specificity	Sensitivity	Specificity
2	OCSTM	44.02±23.27	99.91±00.18	14.72±14.31	98.59±3.08
	OCSVM	13.59±15.12	100.00±0.00	0.00±0.00	100.00±0.00
4	OCSTM	57.57±22.30	99.45±1.59	40.46±21.60	96.60±8.32
	OCSVM	35.13±16.14	100.00±0.00	9.02±14.79	99.50±1.50
6	OCSTM	72.79±17.21	99.31±1.27	60.75±16.23	95.66±6.18
	OCSVM	52.04±22.72	99.67±1.23	36.92±20.99	98.28±1.56
8	OCSTM	73.68±17.28	99.16±1.41	65.87±14.86	94.60±8.80
	OCSVM	62.21±19.58	99.70±0.80	46.68±19.12	97.71±1.60
10	OCSTM	76.47±16.48	98.62±2.30	71.69±14.21	94.78±5.53
	OCSVM	68.66±19.15	99.47±1.07	58.95±13.99	97.36±1.67

Table 5 summarized the averaged percentage of sensitivity and specificity of 50 times simulations, on various training sample sizes of BREAST-CANCER dataset. The averaged specificities of the two algorithms are very close to each other, while the averaged sensitivities of OCSTM are significantly different from those of OCSVM, especially when the training samples are small. When training sample size is 2, the OCSTM has the sensitivities of 44.02 and 14.72 compared to 13.59 and 0.00 for OCSVM, while the corresponding specificities are approximately equal to each other. These experiments validate that OCSTM has significant advantages on avoiding overfitting problem.

4.3 Experiments on all vector datasets

Due to the experiments on BREAST-CANCER dataset, we can conclude that both linear classifiers and nonlinear classifiers have their special application areas. In this part, we evaluate the four algorithms: OCSTM, LOCSTM, OCSVM, and LOCSVM on all 8 vector datasets shown in Table 1. In a spirit similar to the experiments on BREAST-CANCER dataset, we focus on small training sets and concentrate on test accuracy and AUC with various training sample sizes. Since there are only 10 samples in class 3 of LUNG dataset, we introduce four small training sample sizes (2, 4, 6 and 8). As the same reason, the results are averaged over 10 random splits on LUNG dataset, while 50 random splits on other datasets.

In order to avoid unnecessary repetition, we focus on an overall view in this subsection. The averaged percentages of test accuracy (shorted as ACC) and AUC of the four classifiers on various datasets, and their standard deviations are reported in Table 6, while the training sets only have 2 samples. We bold the best test accuracy and AUCs, and in short, tensor-based classifiers have significant performance in comparison with vector-based classifiers.

Besides, paired comparisons of linear and nonlinear classifiers are shown in Table 6 as

Table 6. Averaged percentages of test accuracy and AUCs on various datasets, with 2 training samples.

Dataset	Target Class	metrics	OCSTM (I)	OCSVM (II)	I VS II	LOCSTM (III)	LOCSVM (IV)	III VS IV
BREAST-CANCER	2	ACC	63.64	43.92	>	73.74	68.43	>
		AUC	99.32	99.48	~	99.51	99.48	~
	4	ACC	69.40	65.20	>	65.85	65.83	~
		AUC	84.62	80.93	>	77.55	76.64	>
IRIS	1	ACC	79.85	75.15	>	86.14	84.35	>
		AUC	99.65	99.57	~	99.98	99.97	~
	2	ACC	77.70	75.00	>	55.80	57.38	<
		AUC	93.47	93.43	~	59.29	61.59	<
IMPORT	3	ACC	77.05	73.68	>	84.00	83.64	~
		AUC	89.85	89.75	~	99.00	98.97	~
	1	ACC	54.97	45.22	>	50.42	47.36	>
		AUC	67.23	65.57	>	64.39	65.70	<
IONOSPHERE	1	ACC	52.87	36.69	>	54.24	51.01	>
		AUC	75.36	79.70	<	74.89	72.86	>
	2	ACC	59.67	63.33	<	37.33	47.67	<
		AUC	55.25	55.17	~	--	--	--
LUNG	3	ACC	73.33	73.00	~	58.67	64.67	<
		AUC	64.55	62.73	>	54.15	51.08	>
	1	ACC	54.43	53.88	>	59.91	58.81	>
		AUC	59.81	59.20	>	67.36	66.65	>
DELFTPUMP AR	2	ACC	76.56	73.96	>	73.09	74.06	<
		AUC	85.04	82.38	>	76.57	78.54	<
	2	ACC	93.66	86.24	>	91.66	89.30	>
		AUC	99.75	99.47	~	99.82	99.83	~

well. As can be seen, the test accuracy of OCSTM is better than OCSVM in 10 out of 12 comparisons, and similar in one comparison. And the test accuracy of LOCSTM is better than LOCSVM in 6 comparisons, not much worse in 2 comparisons and similar in 2 comparisons. With the paired comparisons of the AUCs, we can figure that OCSTM is better or similar to OCSVM in 11 out of 12 comparisons, and LOCSTM is better or similar to LOCSVM in 8 out of 12 comparisons. In target class 2 of LUNG dataset, the AUC of LOCSTM and LOCSVM are both under 0.5, which means the classification results do not make any sense. Thus, we use '---' in Table 6, which means the algorithms are invalidation in the case.

For more details about the performance with the training sample sizes, we tested OCSTM, LOCSTM, OCSVM and LOCSVM over different training sizes (2, 4, 6 and 8) on all the datasets in Table 1. To get a more visual image of the classification performance with respect to the training set size, we illustrate the experimental results in Figure 2-8.

From the tendency of the averaged test accuracy and AUC with respect to the training set sizes shown in the figure, we can summarize that tensor representation classifiers have

(a) target class 1, test accuracy

(b) target class 1, AUC

(c) target class 2, test accuracy

(d) target class 2, AUC

(e) target class 3, test accuracy

(f) target class 3, AUC

Figure 2: Classification performance of OCSTM, LOCSTM, OCSVM and LOCSVM on IRIS dataset with respect to the training sample sizes. (a) and (b) are results of test accuracy and AUC of target class 1, (c) and (d) are results of test accuracy and AUC of target class 2, (e) and (f) are results of test accuracy and AUC of target class 3, respectively.

(a) target class 1, test accuracy

(b) target class 1, AUC

Figure 3: Classification performance of OCSTM, LOCSTM, OCSVM and LOCSVM on IMPORT dataset with respect to the training sample sizes. (a) and (b) are results of test accuracy and AUC of target class 1, respectively.

(a) target class 1, test accuracy

(b) target class 1, AUC

Figure 4: Classification performance of OCSTM, LOCSTM, OCSVM and LOCSVM on IONOSPHERE dataset with respect to the training sample sizes. (a) and (b) are results of test accuracy and AUC of target class 1, respectively.

(a) target class 2, test accuracy

(b) target class 2, AUC

(c) target class 3, test accuracy

(d) target class 3, AUC

Figure 5: Classification performance of OCSTM, LOCSTM, OCSVM and LOCSVM on LUNG dataset with respect to the training sample sizes. (a) and (b) are results of test accuracy and AUC of target class 2, (c) and (d) are results of test accuracy and AUC of target class 3, respectively.

(a) target class 1, test accuracy

(b) target class 1, AUC

Figure 6: Classification performance of OCSTM, LOCSTM, OCSVM and LOCSVM on SONAR dataset with respect to the training sample sizes. (a) and (b) are results of test accuracy and AUC of target class 1, respectively.

(a) target class 2, test accuracy

(b) target class 2, AUC

Figure 7: Classification performance of OCSTM, LOCSTM, OCSVM and LOCSVM on DELDTPUMP AR dataset with respect to the training sample sizes. (a) and (b) are results of test accuracy and AUC of target class 2, respectively.

Figure 8: Classification performance of OCSTM, LOCSTM, OCSVM and LOCSVM on USPS dataset with respect to the training sample sizes. (a) and (b) are results of test accuracy and AUC of target class 2, respectively.

a significant performance in all vector-based dataset. When the training set is small (2 or 4 samples), tensor representation classifiers outperform vector-based ones. Specially, for test accuracy, tensor-based classifiers are better than vector-based ones in 11 out of 12 comparisons. For the AUC evaluation criterion, tensor-based classifiers are better than or similar to the vector-based ones in 10 out of 12 comparisons.

5 Experimental evaluations on tensor-based dataset

The ORL dataset [34] consists of forty people’s face images, with ten different grayscale images for each one. All the images are taken against a dark homogeneous background with the head in an upright and frontal position. The 40 subjects have a wide range of facial features, including varying lighting, smiling, closed eyes, skin tone, baldness, beard, glasses and gender. Each image is size of 28×23 with 256 grayscale levels per pixel, and all the features are scaled to $[0, 1]$. Since we are interested in testing the effectiveness of tensor-based algorithms when the dimensionality of the data is large and the available training set is small, we do not perform cropping or resizing of the images which reduces the number of features in the data.

We compare the four classifiers: OCSTM, OCSVM, LOCSTM and LOCSVM for all the 40 target classes of ORL dataset. For each one-class classification experiment, we consider a subset (2 samples, 5 samples and 8 samples) of the target class for training, the rest is considered for testing. There are 120 experiments in all.

Obviously, it is an unbalanced classification problem, since there are only 10 samples in the target class and 390 samples in the outlier. Thus, we introduce two evaluation

indexes: TPR and Gmeans. TPR is the true positive rate of the classifier, and Gmeans report the geometric mean of TPR and TNR (true negative rate). We use Leave One Out (LOO) to find the best parameters in this part, since the target classes are very small.

In a brief exhibition, Table 7 only summarizes the results of OCSTM versus OCSVM on 40 target classes, while training size is 8. We can see that in the 40 experiments, the TPR of OCSTM is much better than that of OCSVM in 12 comparisons, and the rest 28 comparisons are the same. The Gmeans of OCSTM make a significant performance as well. 12 out of 40 comparisons show better performance and 22 out of 40 comparisons are the similar to those of OCSVM. By contrary, for the AUCs of OCSTM, only 4 out of 40 comparisons show better performance than those of OCSVM.

For a full picture of the four classifiers on the 120 experiments, we conclude the TPR, AUC and Gmeans of the two paired comparisons (OCSTM vs OCSVM, LOCSTM vs LOCSVM) in Table 8. We can see that for TPR index, tensor-based classifiers achieved better performance than vector-based classifiers. For Gmeans index, OCSTM performs better than OCSVM. As for AUCs, vector-based classifiers are superior to tensor-based classifiers.

Furthermore, the average performance of the 40 target classes with respect to the training sample sizes is illustrated in Figure 9. We can see that linear one-class classifiers are much better than the nonlinear ones on TPR and Gmeans, and nonlinear one-class classifiers are better on AUC. As can be seen, the tensor-based classifiers are almost better than the corresponding vector ones on TPR and Gmeans. Although tensor-based classifiers have not significant performance on AUC, we can see that the AUCs of OCSTM are almost same as those of OCSVM.

6 The General Algorithm for One-Class Support Tensor Machine

The OCSTM algorithm described above takes second order, i.e., matrices, as input data. Furthermore, we briefly describe a general algorithm for OCSTM in this section. Let $\{\mathbf{T}_i\}, i = 1 \dots l$ denote the training sample set, where $\mathbf{T}_i \in \mathbb{R}^{n_1} \otimes \dots \otimes \mathbb{R}^{n_k}$ is the k -th order tensor.

The optimization problem of OCSTM with k -th order tensors is:

$$\begin{aligned} \min_{\mathbf{W}, \rho, \xi} \quad & \frac{1}{2} \|\mathbf{W}\|^2 + \frac{1}{\nu l} \sum_{i=1}^l \xi_i - \rho \\ \text{s.t.} \quad & (\mathbf{W} \cdot \mathbf{T}_i) \geq \rho - \xi_i \\ & \xi_i \geq 0, i = 1, \dots, l \end{aligned} \tag{36}$$

where $\mathbf{W} \in \mathbb{R}^{n_1} \otimes \dots \otimes \mathbb{R}^{n_k}$ is the weight tensor of the separating hyperplane, ξ_i and ρ have the same meaning as the discussion above.

Table 7. Classification performance of 40 target classes in ORL dataset, with 8 training samples.

Target Class	Algorithm	TPR	AUC	Gmeans	Target Class	Algorithm	TPR	AUC	Gmeans
1	OCSTM	1	1.0000	1.0000	21	OCSTM	1	1.0000	1.0000
	OCSVM	0	0.8359	0.0000		OCSVM	0.5	1.0000	0.7071
2	OCSTM	1	0.9423	0.9687	22	OCSTM	1	1.0000	1.0000
	OCSVM	0.5	1.0000	0.7071		OCSVM	0	1.0000	0.0000
3	OCSTM	1	0.9090	0.8872	23	OCSTM	0	1.0000	0.0000
	OCSVM	0.5	0.9910	0.7071		OCSVM	0	1.0000	0.0000
4	OCSTM	0.5	1.0000	0.7071	24	OCSTM	0.5	0.9949	0.7071
	OCSVM	0	1.0000	0.0000		OCSVM	0.5	0.9936	0.7071
5	OCSTM	0.5	0.9974	0.7062	25	OCSTM	0.5	1.0000	0.7071
	OCSVM	0	1.0000	0.0000		OCSVM	0.5	1.0000	0.7071
6	OCSTM	0	1.0000	0.0000	26	OCSTM	0.5	1.0000	0.7071
	OCSVM	0	0.0000	0.0000		OCSVM	0.5	1.0000	0.7071
7	OCSTM	0.5	1.0000	0.7071	27	OCSTM	1	1.0000	1.0000
	OCSVM	0.5	1.0000	0.7071		OCSVM	1	1.0000	1.0000
8	OCSTM	0.5	0.9397	0.6859	28	OCSTM	0.5	0.9782	0.6971
	OCSVM	0.5	1.0000	0.7071		OCSVM	0.5	0.9551	0.7062
9	OCSTM	0	0.9718	0.0000	29	OCSTM	1	0.9846	0.9740
	OCSVM	0	1.0000	0.0000		OCSVM	0	1.0000	0.0000
10	OCSTM	0	0.7641	0.0000	30	OCSTM	0.5	1.0000	0.7071
	OCSVM	0	0.9513	0.0000		OCSVM	0.5	1.0000	0.7071
11	OCSTM	0.5	0.9987	0.7062	31	OCSTM	0	1.0000	0.0000
	OCSVM	0.5	1.0000	0.7071		OCSVM	0	1.0000	0.0000
12	OCSTM	0.5	0.9974	0.7071	32	OCSTM	0.5	1.0000	0.7071
	OCSVM	0.5	1.0000	0.7071		OCSVM	0.5	1.0000	0.7071
13	OCSTM	0.5	1.0000	0.7071	33	OCSTM	1	1.0000	1.0000
	OCSVM	0.5	1.0000	0.7071		OCSVM	0.5	1.0000	0.7071
14	OCSTM	0	1.0000	0.0000	34	OCSTM	0.5	1.0000	0.7071
	OCSVM	0	1.0000	0.0000		OCSVM	0.5	1.0000	0.7071
15	OCSTM	0.5	1.0000	0.7071	35	OCSTM	0	0.7090	0.0000
	OCSVM	0.5	1.0000	0.7071		OCSVM	0	0.9885	0.0000
16	OCSTM	0	0.6141	0.0000	36	OCSTM	0	0.6885	0.0000
	OCSVM	0	0.9295	0.0000		OCSVM	0	0.9885	0.0000
17	OCSTM	1	0.9641	0.9780	37	OCSTM	1	1.0000	1.0000
	OCSVM	1	1.0000	1.0000		OCSVM	1	1.0000	1.0000
18	OCSTM	1	1.0000	1.0000	38	OCSTM	1	1.0000	0.9987
	OCSVM	0.5	1.0000	0.7071		OCSVM	1	1.0000	1.0000
19	OCSTM	1	0.9872	0.9793	39	OCSTM	0.5	0.9974	0.7062
	OCSVM	0.5	1.0000	0.7071		OCSVM	0.5	1.0000	0.7071
20	OCSTM	0	0.9397	0.0000	40	OCSTM	1	0.9923	0.9936
	OCSVM	0	0.9974	0		OCSVM	0.5	0.9974	0.7071

Table 8. Paired Comparisons with the different training samples.

Training Sample	Algorithm Comparison	TPR			AUC			Gmeans		
		>	~	<	>	~	<	>	~	<
2	OCSTM vs OCSVM	19	20	1	12	1	27	19	20	1
	LOCSTM vs LOCSVM	26	14	0	0	0	40	18	2	20
5	OCSTM vs OCSVM	26	9	5	8	7	25	26	8	6
	LOCSTM vs LOCSVM	15	25	0	0	0	40	11	1	28
8	OCSTM vs OCSVM	12	28	0	4	19	17	12	22	6
	LOCSTM vs LOCSVM	9	31	0	0	1	39	9	0	31

(a) TPR

(b) AUC

(c) Gmeans

Figure 9: Averaged performance of the 40 target classes on ORL dataset with respect to the training sample sizes. (a), (b) and (c) are the averaged results of TPR, AUC and Gmeans, respectively.

First, we introduce Lagrange multipliers $\alpha_i, \beta_i \geq 0, i = 1 \dots l$ for each of the inequality constrains. This gives Lagrangian:

$$\begin{aligned} \mathcal{L}(\mathbf{W}, \rho, \xi, \alpha, \beta) = & \frac{1}{2} \|\mathbf{W}\|^2 + \frac{1}{\nu l} \sum_{i=1}^l \xi_i - \rho \\ & - \sum_{i=1}^l \alpha_i ((\mathbf{W} \cdot \mathbf{T}_i) - \rho + \xi_i) - \sum_{i=1}^l \beta_i \xi_i. \end{aligned} \quad (37)$$

Setting the derivatives with respect to the primal variables $\mathbf{W}, \xi_i, \rho$ equal to zero, we have:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = 0 \Rightarrow \mathbf{W} = \sum_{i=1}^l \alpha_i \mathbf{T}_i \quad (38)$$

$$\frac{\partial \mathcal{L}}{\partial \rho} = 0 \Rightarrow \sum_{i=1}^l \alpha_i = 1 \quad (39)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow \frac{1}{\nu l} - \alpha_i - \beta_i = 0 \quad (40)$$

Then we can get the dual problem:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j (\mathbf{T}_i \cdot \mathbf{T}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu l} \\ & \sum_{i=1}^l \alpha_i = 1, i = 1, \dots, l \end{aligned} \quad (41)$$

Obviously, the optimization problem in (41) is the generalization of the standard linear OCSVM to tensor patterns in tensor space. When the input samples $\mathbf{T}_i$ are vectors, it degenerates into the standard linear OCSVM. By introducing a nonlinear feature mapping $\Phi(\cdot)$, $\mathbf{T}_i$ can be mapped from original tensor space into feature space. With kernel trick, let

$$K(\mathbf{T}_i, \mathbf{T}_j) = (\Phi(\mathbf{T}_i) \cdot \Phi(\mathbf{T}_j)) \quad (42)$$

Then we can get the non-linear OCSTM classification model by solving the convex quadratic programming:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j K(\mathbf{T}_i, \mathbf{T}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu l} \\ & \sum_{i=1}^l \alpha_i = 1, i = 1, \dots, l \end{aligned} \quad (43)$$

The resulting decision function is:

$$f(\mathbf{X}) = \text{sgn}\left(\sum_{i=1}^l \alpha_i K(\mathbf{T}, \mathbf{T}_i) - \rho\right) \quad (44)$$

From the above statement, we can see that tensor-based learning algorithm degenerates into the study of kernel function, and the success of kernel methods depends strongly on the data representation encoded into the kernel function. There are some studies on kernel function for tensors, such as naive kernel [23], matrix kernel function [25, 26], factor kernels [23], K_{3rd} kernel [27] and structure-preserving kernel [24].

7 Conclusions and future work

In this work we propose a new one-class classification algorithm named the One-Class Support Tensor Machine. OCSTM uses tensor as input data, and aims to separate almost all the samples of target class from the origin with maximal margin. The benefits of the proposed algorithm are twofold. First, the use of direct tensor representation is helpful to retain the data topology more efficiently. The second benefit is that tensor representation can greatly reduce the number of parameters. It helps overcome the overfitting problem encountered mostly in vector-based algorithms and especially suits for high dimensional and small sample size problem. To solve the optimization problem corresponding to OCSTM, we use alternative projection method in which the parameters corresponding to the projections are estimated by solving a standard OCSVM optimization problem. For the above mentioned benefits of tensor representation, we validate our proposed algorithm both on vector-based and tensor-based datasets. As expected, the tensor-based one-class classification algorithms yield better generalization performance.

However, there are some drawbacks of the proposed method. Although tensor representation has greatly reduced the number of parameters, the iteration of solving parameters still cost lots of time. Due to the alternative projection algorithm, the training time of OCSTM is much more than vector-based algorithms. A possible direction on the work is to investigate more efficient computational methods for solving the optimization problems of OCSTM. Further study on this topic will also include the kernel technology. We use TRBF kernel matrix, which is based on the Gauss kernel function in vector space, for nonlinear classification problems in this paper. Further study on kernel technology and the application of high order OCSTM will be dealt with in the future. Another interesting topic is to apply the OCSTM to real world classification, since the data point is originally expressed in tensor representation in many application areas.

Acknowledgments

The work is supported by the National Science Foundation of China (Grant No. 11171346) and the "New Start" Academic Research Projects of Beijing Union University (Zk10201513). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

- [1] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, USA, (1995).
- [2] B. Scholkopf, J.C. Platt, J. S. Taylor, A. J. Smola, R. C. Williamson, Estimating the support of a high-dimensional distribution, Technical report, Microsoft Research, MSR-TR-99-87(1999).
- [3] D. M. J. Tax, R. P. W. Duin, Support vector domain description, Pattern Recognition Letter, 20 (11)(1999): 1191-1199.
- [4] W. Zhu, P. Zhong, A new one-class SVM based on hidden information. Knowledge-Based Systems 60(2014): 35-43.
- [5] Z. Zhao, P. Zhong, Y. Zhao, Reduced least squares one-class SVM in empirical feature space for imbalanced data, ICIC Express Letters, 5(11)(2010): 4115-4121.
- [6] L. M. Manevitz, M. Yousef. One-class SVMs for document classification. Journal of Machine Learning Research 2(2001):139-154.
- [7] Y. Chen, X. Zhou, T. S. Huang. One-class SVM for learning in image retrieval. In Proc. IEEE Int'l Conf. on Image Processing, Thessaloniki, Greece (2001).
- [8] K. Etemad, R. Chellappa, Discriminant analysis for recognition of human face images, Journal of the Optical Society of America A: Optics Image Science and Vision, 14(8)(1997):1724-1733.
- [9] R. D. Green, 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, 14(2)(2004): 191-198.
- [10] H. Lu, K. N. Plataniotis, A. N. Venetsanopoulos, MPCA: multilinear principal component analysis of tensor objects, IEEE Transactions on Neural Networks, 19(1)(2008): 18-39.
- [11] N. Renard, S. Bourennane, Dimensionality reduction based on tensor modeling for classification methods, IEEE Transactions on Geoscience and Remote Sensing, 47(4)(2009): 1123-1131.
- [12] D. Tao, X. Li, X. Wu, S. J. Maybank, Supervised tensor learning. Knowledge and Information Systems, 13(1)(2007): 1-42.

- [13] X. Wu, S. J. Maybank, W. Hu, X. Li, D. Tao, Supervised tensor learning, IEEE International Conference on Data Mining, IEEE Computer Society, 13(2005):450-457.
- [14] D. Cai, X. Hei, J. Han, Learning with tensor representation, Department of Computer Science, University of Illinois at Urbana-Champaign, UIUCDCS-R-2006-2716 (2006)
- [15] D. Cai, X. He, J. Wen, J. Han, W. Ma, Support tensor machines for text categorization, Technical report, Computer Science Department, UIUCDCS-R-200-2714 (2006).
- [16] C. Cortes, V. Vapnik, Support vector networks, Machine Learning, 20(3)(1995): 273-297.
- [17] B. Scholkopf, A. J. Smola, R. C. Williamson, New support vector algorithms, Neural Computation, 12(5)(2000): 1207-1245.
- [18] X. Zhang, X. Gao, Y. Wang, Twin support tensor machines for MCS detection, Journal of Electronics(China), 26(3)(2009): 318-325.
- [19] R. Khemchandani, A. Karpatne, S. Chandra, Proximal support tensor machines, Int. J. Mach. Learn. & Cyber, 4(2013):703-712.
- [20] I. Kotsia, W. Guo, I. Patras, Higher rank support tensor machines for visual recognition, Pattern Recognition, 45(12)(2012): 4192-4203.
- [21] Z. Hao, L. He, B. Chen, X. Yang, A linear support higher-order tensor machine for classification, IEEE Transactions on Image Processing 22(7)(2013): 2911-2920.
- [22] X.Liu, T. Guo, L. He, X.Yang, A low-rank approximation-based transductive support tensor machine for semisupervised classification, Image Processing, IEEE Transactions on, 24(6)(2015): 1825-1838.
- [23] M. Signoretto, L. D. Lathauwerb, J. A. K. Suykens, A kernel-based framework to tensorial data analysis, Neural Networks 24 (2011): 861-874.
- [24] L. He, X. Kong, P. S. Yu et.al, DuSK: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages, Proc SIAM Int Conf Data Min (2014): 127-135.
- [25] C. GAO, X. WU, Kernel support tensor regression, Procedia Engineering 29(2012): 3986 - 3990.
- [26] P. Daniusis, P. Vaitkus, Kernel regression on matrix patterns, Lithuanian Mathematical Journal, Spec. edition 48-49(2008): 191-195.
- [27] S. Park, Multifactor analysis for fMRI brain image classification by subject and motor task, Electrical and computer engineering technical report, Carnegie Mellon University (2011).
- [28] S. Msji, A. Berg, J. Malik. Efficient classification for additive kernel SVMs. IEEE Trans. Pattern Anal. Mach. Intell., 1(35)(2013): 66-77.
- [29] C. Chang, C. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology, 2(27)(2011): 1-27.

- [30] <http://prlab.tudelft.nl/users/david-tax>.
- [31] A. Airola, T. Pahikkala, W. Waegeman, B. D. Baets, T. Salakoski, An experimental comparison of cross-validation techniques for estimating the area under the ROC curve, *Comput.Stat. Data Anal.* 55 (4)(2011): 1828-1844.
- [32] H. Han, Analyzing support vector machine overfitting on microarray data, *Intelligent Computing in Bioinformatics* (2014): 148-156.
- [33] C. Y. Chang, M. T. Hsu, E. X. Esposito, Y. J. Tseng, Oversampling to overcome overfitting: exploring the relationship between data set composition, Molecular Descriptors, and Predictive Modeling Methods, *Journal of Chemical Information and Modeling* 53(2013): 958-971.
- [34] AT&T Labs Cambridge, The Olivetti and Oracle Research Laboratory database of faces (1994), <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>.