International Journal of Hybrid Information Technology
Vol. 9, No.11 (2016), pp. 381-390
http://dx.doi.org/10.14257/ijhit.2016.9.11.32

A Kernel-based Matrixzed One-Class Support Vector Machine

Yanyan Chen*, June Yuan and Zhengkun Hu

College of Applied Science and Technology, Beijing Union University, Beijing 100101, China
**Corresponding author. E-mail: yykjtyanyan@buu.edu.cn*

Abstract

One-class support vector machine (OCSVM) is an important and efficient classifier used when only one class of data is available while others are too expensive or difficult to collect. It uses vector as input data, and trains a linear or nonlinear decision function in vector space. However, the traditional vector-based classifiers may fail when input is matrix. Therefore, it makes sense to study matrixzed classifiers which can make use of the structural information presented in the data. In this paper we propose a matrix-based one-class classification algorithm named Kernel-based Matrixzed One-class Support Vector Machine (KMatOCSVM). It aims to convert the OCSVM to suit for matrix representation data and to deal with nonlinear one-class classification problems. The efficiency and validity of the proposed method is illustrated by four real-world matrix-based human face datasets.

Keywords*: One-class support vector machine; Matrix pattern; Kernel-based method; One-class classification problem*

1. Introduction

One-class support vector machine (OCSVM) [1, 2] is an important and widely used classifier. It can be used when the negative samples difficult to collected or labeled, such as intrusion detection, fault detection and diagnosis, and the classification of remote sensing data [3, 4]. Recent years have witnessed more and more theoretical studies and applications on OCSVM [5, 6], and all of them use vector as input.

In practice, however, many of the original objects are represented as matrix, *i.e.* gray face image [7, 8]. The traditional vector-based one-class classification methods represented by OCSVM cannot directly solve the problems when matrix is considered as input data. Although there are some methods which can convert matrix directly into vector, they may cause data correlation damage and structural information loss [9, 10]. All these reasons lead researchers to consider using matrix as input and develop corresponding learning algorithms for one-class classification problems.

Recently, there are lots of researches on converting vector-based algorithms to corresponding matrix patterns. Yang *et al*. proposed a two-dimensional principal component analysis (2DPCA) which extracted features directly from the matrix [11]. Chen *et al*. developed a more general principal component analysis (MatPCA) to extract features based on matrix pattern [12]. Wang and Chen proposed a matrixized least squares support vector machine (MatLSSVM) that could directly classify the objects represented by matrix [13]. Furthermore, Wang *et al*. improved MatLSSVM and proposed an efficient kernelized classifier named kernel-based matrixized least square support vector machine (KMatLSSVM) [14]. Yan *et al*. [15] developed a new variant of OCSVM which directly took matrix as input, and named it Matrix-Pattern-Oriented One-class SVM (MatOCSVM). All the experiment results verify that when matrix is used as input, the matrixized classifier has a superior classification performance to its vector version.

ISSN: 1738-9968 IJHIT

In this paper, we propose a nonlinear one-class classifier which directly takes matrix as input. The new algorithm is intended to solve nonlinear classification problems which cannot be dealt with by MatOCSVM. It is named Kernel-based Matrixzed One-Class Support Vector Machine (KMatOCSVM). KMatOCSVM, directly using matrix as input, can help to retain the data topology efficiently in comparison with vector-based classifier. The efficiency and validity of the proposed method is illustrated by real-world matrix-based human face datasets.

The rest of this paper is organized as follows. A brief summary of some relevant concepts on the standard OCSVM and MatOCSVM are presented in Section 2. The proposed KMatOCSVM is described in Section 3 and the experimental evaluation is presented in Section 4. Finally, conclusions of our work are drawn in Section5.

2. Relevant Researches

Before presenting our work, we first briefly review the algorithms of standard OCSVM and linear MatOCSVM which are relevant to our research in this section.

2.1. One-Class Support Vector Machine

One-class support vector machine is a useful technique for data classification. It aims to learn a single class by determining a decision function with maximal margin from the origin that contains almost all the data of the target class. In this subsection we only review the algorithm of standard linear OCSVM.

Consider training data $\mathbf{x}_i \in \square^n, i=1,...,l$, the decision hyperplane relative to the membership of the observation $\mathbf{x}$ to the considered class is given by:

$$f(\mathbf{x}) = sgn\left((\mathbf{w}\cdot\mathbf{x}) - \rho\right), \tag{1}$$

where parameters $\mathbf{w}$ and ρ result from the optimization problem:

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

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

In practice, the quadratic program (2) is solved via its dual:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j) \\ s.t. \quad & 0 \leq \alpha_i \leq \frac{1}{\nu l} \\ & \sum_{i=1}^{l}\alpha_i = 1, i=1,...,l \end{aligned} \tag{3}$$

The optimal normal vector is given by $\mathbf{w} = \sum_{i=1}^{l}\alpha_i\mathbf{x}_i$, where α_i is the solution of the dual problem (3) and training samples $\mathbf{x}_i$ with non-zero α_i are support vectors.

2.2. Matrixized One-Class Support Vector Machine

MatOCSVM is a matrix-pattern-oriented version of OCSVM and can directly classify the samples represented by matrix.

Given a set of training samples $\{\mathbf{X}_i\}, i=1\ldots l$, each of the training sample $\mathbf{X}_i \in \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2}$ is a matrix data point, where $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$ are two vector spaces.

The discriminant function of MatOCSVM is formed as

$$f(\mathbf{X}) = sgn\left(\mathbf{u}^T \mathbf{X}\mathbf{v} - \rho\right), \tag{4}$$

where $\mathbf{u} \in \mathbb{R}^{n_1}$ and $\mathbf{v} \in \mathbb{R}^{n_2}$ are both the weight vectors and ρ is a bias.

The optimization problem of MatOCSVM is:

$$\begin{aligned} &\min_{\mathbf{u}\in\mathbb{R}^{n_1},\mathbf{v}\in\mathbb{R}^{n_2},\xi\in\mathbb{R}^{l},\rho\in\mathbb{R}} \frac{1}{2}\|\mathbf{u}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i - \rho \\ &s.t.\ \ \mathbf{u}^T\mathbf{X}_i\mathbf{v} \geq \rho - \xi_i \\ &\xi_i \geq 0, i = 1,\ldots,l \end{aligned} \tag{5}$$

The decision variables in quadratic program problem (5) can be solved in an iteration way, and more elaborate description about MatOCSVM can be found in [15].

3. Kernel-based Matrixized One-Class Support Vector Machine

Since our Kernel-based MatOCSVM is a kernelized version of MatOCSVM, the optimization problem is the same as that of (5).

To solve the problem, Lagrangian function corresponding to KMatOCSVM is:

$$L(\mathbf{u},\mathbf{v},\xi,\rho,\alpha,\beta) = \frac{1}{2}\|\mathbf{u}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i - \rho - \sum_{i=1}^{l}\alpha_i\left(\mathbf{u}^T\mathbf{X}_i\mathbf{v} - \rho + \xi_i\right) - \sum_{i=1}^{l}\beta_i\xi_i , \tag{6}$$

where $\alpha_i, \beta_i \geq 0, i = 1,...,l$ are Lagrange multipliers, for each of the inequality constrains.

According to the KKT necessary and sufficient optimality conditions, we differentiate the primal variables $\mathbf{u}, \mathbf{v}, \xi_i, \rho$,

$$\frac{\partial L}{\partial \mathbf{u}} = \mathbf{u} - \sum_{i=1}^{l}\alpha_i\mathbf{X}_i\mathbf{v}, \tag{7}$$

$$\frac{\partial L}{\partial \mathbf{v}} = -\sum_{i=1}^{l}\alpha_i\mathbf{X}_i^T\mathbf{u}, \tag{8}$$

$$\frac{\partial L}{\partial \xi_i} = \frac{1}{\nu l} - \beta_i - \alpha_i, \tag{9}$$

$$\frac{\partial L}{\partial \rho} = \sum_{i=1}^{l}\alpha_i - 1, \tag{10}$$

From equation (7) we can see that $\mathbf{u}$ and $\mathbf{v}$ are dependent on each other, and they cannot be solved independently. Hence, we resort to the iteration method to solve this optimization problem. The specific method can be described as follows.

First we fix $\mathbf{v}$ and let $\mathbf{x}_i = \mathbf{X}_i\mathbf{v}$. According to (5), we can construct the optimal quadratic programming problem to solve $\mathbf{u}$:

$$\begin{aligned} &\min_{\mathbf{u}\in\mathbb{R}^{n_1},\xi\in\mathbb{R}^{l},\rho\in\mathbb{R}} \frac{1}{2}\|\mathbf{u}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i - \rho \\ &s.t.\ \ (\mathbf{u}\cdot\mathbf{x}_i) \geq \rho - \xi_i \\ &\xi_i \geq 0, i = 1,\ldots,l \end{aligned} \tag{11}$$

It can be seen that the optimization problem (11) is similar in structure to that of standard OCSVM (2). Thus the weight vector **u** can be solved by using the same approach as that of OCSVM.

In the original MatOCSVM algorithm, the optimization problem (11) is solved via its dual problem (3). To achieve the kernelization, we directly introduce a kernel-based function instead of the original inner product $\left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$ in dual problem (3). Thus the kernelization dual problem can be described as:

$$\begin{aligned} \min_{\alpha} \quad & \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j k\left(\mathbf{x}_i,\mathbf{x}_j\right) \\ s.t. \quad & 0 \le \alpha_i \le \frac{1}{\nu l} \\ & \sum_{i=1}^{l}\alpha_i = 1, i = 1,...,l \end{aligned} \tag{12}$$

where $k(\cdot\,,\,\cdot)$ is the kernel-based function such as $k\left(\mathbf{x}_i,\mathbf{x}_j\right)=\mathbf{e}^{\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2/2\sigma^2}$ [16]. Solving (12) determines the Lagrangian multipliers α_i^*, then we can obtain $\mathbf{u}=\sum_{i=1}^{l}\alpha_i^*\mathbf{x}_i$.

In this kernel-induced procedure, the right weight vector **v** can be viewed as a transformed operator. The original space spanned by vector **Xv** is mapped into a feature space through the kernel function $k(\cdot\,,\,\cdot)$. That leads to a kernelized MatOCSVM.

When left weight vector **u** is obtained by solving (12), we can calculate the right weight vector **v** by gradient descent algorithm. According to equation (8), **v** can be updated by:

$$\mathbf{v}_{t+1}=\mathbf{v}_t-\eta\frac{\partial L}{\partial \mathbf{v}_t}=\mathbf{v}_t-\eta\left(-\sum_{i=1}^{l}\alpha_i\mathbf{X}_i^T\mathbf{u}\right), \tag{13}$$

where η is the learning rate and t is the iterative counter.

Thus, **u** and **v** can be obtained iteratively by solving (12) and (13). Consequently, the decision function of the kernelized MatOCSVM for input matrix data point **X** is formed as

$$f\left(\mathbf{X}\right)=sgn\left(\sum_{i=1}^{l}\alpha_i^*\mathbf{v}^T k(\mathbf{X}_i\mathbf{u},\mathbf{X}\mathbf{u})\mathbf{v}-\rho\right), \tag{14}$$

where the parameter ρ is calculated by

$$\rho=mean\left(\sum_{i=1}^{l}\alpha_i^*\mathbf{v}^T k(\mathbf{X}_i\mathbf{u},\mathbf{X}_i\mathbf{u})\mathbf{v}\right). \tag{15}$$

On the whole, KMatOCSVM is made up of two procedures. First, the solution of right weight vector **v** can be obtained by gradient descent algorithm. Second, for every fixed **v**, the optimization problem with left weight vector **u** can be solved by a kernelization dual problem. The full algorithm of KMatOCSVM is summarized in Table 1, wiht $\mathbf{v}_0$ being the initial value of **v**, *maxIter* denoting the maximal iterative count and ε being the convergence condition.

Table 1. The Algorithm of KMatOCSVM

Input: The training samples $\mathbf{X}_i \in \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2}$ **.**

Output: The parameters in decision plane: $\mathbf{u}^* \in \mathbb{R}^{n_1}$**,** $\mathbf{v}^* \in \mathbb{R}^{n_2}$ **and** $\rho \in \mathbb{R}$ **.**

Step 1 Initialization $\mathbf{v}_0$, η , *maxIter* , ε , ν **and** σ **.**

Step 2 Obtain u by solving the optimization problems (12), where $\mathbf{x}_i = \mathbf{X}_i \mathbf{v}$ **.**

Step 3 update v through (13).

Step 4 Do step 2 and step 3 iteratively until convergence: If the iteration number exceeds the *maxIter* number of iteration or the below convergence condition is satisfied:

$$\left\| \mathbf{v}_i - \mathbf{v}_{i-1} \right\| \le \varepsilon$$

Step 5 End.

4. Experimental Evaluation

4.1. Description of Datasets

In this section, we evaluate the performance of KMatOCSVM with experiments on 4 publicly available image datasets which are presented as a matrix. Description of these datasets which are obtained on [17] can be seen in Table 2.

Table 2. Information of the Various Dataset

Dataset	#Samples	#Class	#Attr
ORL	400	40	28×23
PIE	1428	68	32×32
C80	1145	80	32×32
YALE	165	15	100×100

To get a visual image of these human face datasets, ORL datasets for example, several typical images are illustrated in Figure 1. We can see that these human faces are gray images with 256 grayscale levels per pixel. The subjects selected have a wide range of facial features, including skin tone, baldness, beard and gender.

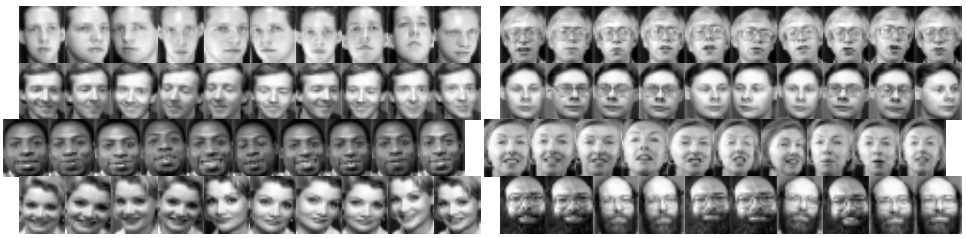

Figure 1. Several Typical Images of ORL Datasets

Since we are interested in testing the effectiveness of KMatOCSVM when the data is represented as matrix, we do not perform cropping or resizing of the images which reduce the number of features in the data. All the features in these four datasets are scaled to [0,1].

4.2. Preparation of Experiments

For one class classification problems, we consider each person as a target class. For simplicity, 10 target classes from each dataset are randomly chosen in all our

experiments due to the fact that there exists too many target classes. To verify the effectiveness of the algorithms, the size of training sets in our experiments is assigned to 6. For each target class, we randomly choose 6 samples as training samples, and combine the remaining samples in the target class and all the outliers to form the testing set. For statistical significance, the results are averaged over 10 random splits from target class and the average performance is reported.

Furthermore, we use RBF kernel $k\left(\mathbf{x}_i,\mathbf{x}_j\right)=\mathbf{e}^{\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2/2\sigma^2}$ in the experiments, for it has been demonstrated that RBF kernel usually outperforms other kernels [18]. We consider AUC, the area under the ROC curve, as the performance metrics for comparisons since the AUC is always used to measure the performance of a one-class classifier [19].

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. There are three tuning parameters: ν, σ and η. The possible choices for these parameters are ν ={0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, σ ={2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{2}, 2^{3}, 2^{4}, 2^{5}} and η ={10^{-1},10^{0},10^{1}}. 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.

4.3. Classification Performance

In this experiment, we compare the performance of KMatOCSVM with that of MatOCSVM on all publicly available datasets shown in Table 2. Table 3 ~ Table 6 summarize the average AUC and time cost of the two algorithms on various datasets. The best results are bolded in each comparison.

For ORL dataset, we can conclude from Table 3 that the AUC of KMatOCSVM is better than that of MatOCSVM in 7 out of 10 comparisons. And the training time of KMatOCSVM is shorter than that of MatOCSVM in all 10 experiments.

Table 3. Averaged AUC and Time Cost on 10 Random Target Classes In ORL Dataset

Target class	AUC		Training Time	
	KMatOCSVM	MatOCSVM	KMatOCSVM	MatOCSVM
Face0 5	**0.8504**	0.7827	**0.1328**	0.1683
Face0 6	0.7845	**0.8286**	**0.1711**	0.1869
Face 07	**0.8827**	0.7985	**0.1343**	0.1780
Face 09	**0.7601**	0.7451	**0.1338**	0.1499
Face 10	0.8016	**0.8549**	**0.1342**	0.1536
Face 13	**0.7644**	0.6879	**0.1410**	0.1589
Face 15	**0.7905**	0.6525	**0.1493**	0.1731
Face 22	**0.9066**	0.8624	**0.1421**	0.1521
Face 25	0.7695	**0.8075**	**0.1397**	0.1743
Face 38	**0.8522**	0.8469	**0.1340**	0.1487

For PIE dataset, as shown in Table 4, the AUC of KMatOCSVM is better than that of MatOCSVM in 9 out of 10 comparisons. The training time of KMatOCSVM is longer than that of MatOCSVM in all 10 experiments, which is opposite to the result of ORL dataset.

Table 4. Averaged AUC and Time Cost on 10 Random Target Classes In PIE Dataset

Target class	AUC		Training Time	
	KMatOCSVM	MatOCSVM	KMatOCSVM	MatOCSVM
Face 05	**0.7320**	0.5583	0.2642	**0.2537**
Face 10	**0.7541**	0.5613	0.2571	**0.2425**
Face 14	**0.7558**	0.6069	0.2649	**0.2487**
Face 23	**0.7342**	0.6194	0.2614	**0.2515**
Face 24	0.7156	**0.7981**	0.2661	**0.2581**
Face 30	**0.7930**	0.6463	0.2604	**0.2481**
Face 40	**0.7492**	0.7039	0.2712	**0.2548**
Face 41	**0.7359**	0.5962	0.2615	**0.2521**
Face 45	**0.7367**	0.6324	0.2612	**0.2449**
Face 68	**0.8265**	0.6545	0.2550	**0.2489**

For C80 dataset, the AUC and training time cost of KMatOCSVM are all better than MatOCSVM in all 10 experiments, as shown in Table 5. For YALE dataset, as shown in Table 6, the AUC of KMatOCSVM is better than that of MatOCSVM only in 5 out of 10 comparisons while the training time of KMatOCSVM are all outstanding.

Table 5. Averaged AUC and Time Cost on 10 Random Target Classes In C80 Dataset

Target class	AUC		Training Time	
	KMatOCSVM	MatOCSVM	KMatOCSVM	MatOCSVM
Face **3**	**0.8084**	0.5437	**0.2843**	0.3266
Face **9**	**0.8297**	0.6523	**0.2889**	0.3552
Face **25**	**0.8508**	0.6095	**0.2855**	0.3463
Face **30**	**0.8578**	0.7028	**0.3212**	0.4002
Face **35**	**0.9023**	0.8370	**0.2749**	0.3738
Face **37**	**0.9419**	0.6379	**0.2787**	0.3367
Face **40**	**0.8911**	0.7835	**0.2783**	0.3850
Face **45**	**0.8823**	0.8903	**0.2793**	0.3259
Face **51**	**0.9313**	0.6691	**0.2702**	0.3182
Face **60**	**0.8235**	0.6917	**0.2645**	0.3155

Table 6. Averaged AUC and Time Cost on 10 Random Target Classes In YALE Dataset

Target class	AUC		Training Time	
	KMatOCSVM	MatOCSVM	KMatOCSVM	MatOCSVM
Face01	0.8704	**0.8761**	**0.2957**	0.3401
Face02	**0.7101**	0.7030	**0.2936**	0.3207
Face03	0.5766	**0.6557**	**0.3057**	0.3518
Face05	0.6238	**0.6458**	**0.3177**	0.3367
Face08	0.5835	**0.6336**	**0.2950**	0.3095
Face09	**0.7108**	0.6610	**0.3149**	0.3707
Face10	**0.7701**	0.7144	**0.3079**	0.3445
Face12	**0.6706**	0.6616	**0.3065**	0.3436
Face14	**0.6834**	0.6370	**0.3012**	0.3484
Face15	0.6831	**0.7532**	**0.3362**	0.3677

Generally speaking, for nonlinear classification problems, kernel-based classifiers often take longer training time to achieve higher accuracy than those of linear classifiers. In contrast, the proposed KMatOCSVM takes shorter training time than that of MatOCSVM based on the experimental results of 3 out of 4 datasets. A possible explanation is because of the iteration method corresponding to KMatOCSVM and MatOCSVM, which leads to the fact that the training time depends on the number of iterations. Therefore, the time cost on calculating the kernel matrix can be negligible. In summary, the proposed kernel-based KMatOCSVM has an outstanding performance compared to the linear MatOCSVM, both on the AUC and the training time.

5. Conclusions and Future Work

In this paper we investigate a new metricized nonlinear one-class classification algorithm named Kernel-based Matrixized One-Class Support Vector Machine, which takes matrix as input data. A significant advantage of KMatOCSVM is the use of direct matrix which is helpful in retaining the data topology efficiently. Our experimental results on publicly available datasets demonstrate that KMatOCSVM performs well in solving on nonlinear classification problems.

However, there are some drawbacks of the proposed method. Since the iteration of solving parameters costs lots of time, the training time of KMatOCSVM is much longer than vector-based OCSVM. A possible direction for future work is to investigate more efficient computational methods to solve the optimization problems of KMatOCSVM. Kernel technology and application of KMatOCSVM can also be dealt with in the future. Another interesting topic is to apply KMatOCSVM to real world classification, for in many application areas the data point is originally expressed in matrix.

Acknowledgments

The work is supported by the “New Start” Academic Research Projections of Beijing Union University (Zk10201513).

References

[1] B. Scholkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola and R.C. Williamson, “Estimating the support of a high-dimensional distribution”, Technical report, Microsoft Research, MSR-TR-99-87, **(1999)**.
[2] D. M. J. Tax and R. P. W. Duin, “Support vector domain description”, Pattern Recognition Letter, vol. 20, no. 11, **(1999)**, pp. 1191–1199.
[3] L. M. Manevitz and M. Yousef, “One-Class SVMs for Document Classification”, Journal of Machine Learning Research, vol. 2, **(2001)**, pp. 139-154.
[4] Y. Chen, X. Zhou and T. S. Huang, “One-class SVM for learning in image retrieval”, In Proc. IEEE Int'l Conf. on Image Processing, Thessaloniki, Greece, **(2001)**.
[5] W. Zhu and P. Zhong, “A new one-class SVM based on hidden information”, Knowledge-Based Systems, vol. 60, (**2014**), pp. 35–43.
[6] Z. Zhao, P. Zhong and Y. Zhao, “Reduced least squares one-class SVM in empirical feature space for imbalanced data”, ICIC Express Letters, vol. 5, no. 11, **(2010)**, pp. 4115-4121.
[7] J. Xia, L. Xiao and L. Wan, “Application of random-fuzzy probability statistics method”, Mathematical Modelling of Engineering Problems, vol. 3, no. 1, **(2016)**, pp. 19-24.
[8] P. Shen, “An improved parallel Bayesian text classification algorithm”, Review of Computer Engineering Studies, vol. 3, no. 1, **(2016)**, pp. 6-10.
[9] D. Tao, X. Li, X. Wu and S. J. Maybank, “Supervised tensor learning”, Knowledge and Information Systems, vol. 13, no. 1, **(2007)**, pp. 1-42.
[10] Y. Chen, K. Wang and P. Zhong, “One-class support tensor machine”, Knowledge - Based Systems, vol. 96, **(2016)**, pp. 14-28.
[11] J. Yang, D. Zhang and A. Frangi, “Two-dimension pca: a new approach to appearance-based face representation and recognition”, IEEE Trans Pattern Anal mach Intell, vol. 26, no. 1, **(2004)**, pp. 131-137.
[12] S. Chen, Y. Zhu, D. Zhang and J. Yang, “Feature extraction approaches based on matrix pattern: matpca and matflda”, Pattern Recogn Lett, vol. 26, **(2005)**, pp. 1157-1167.

[13] Z. Wang and S. Chen, "New least squares support vector machines based on matrix pattern", Neural Process Lett, vol. 26, **(2008)**, pp. 41-56.
[14] Z. Wang, X. He, D. Gao and X. Xue, "An efficient kernel-based matrixized least squares support vector machine", Neural Comput and Applic, vol. 22, **(2013)**, pp. 143-150.
[15] Y. Yan, Q. Wang, G. Ni, Z. Pan and R. Kong, "One-class support vector machines based on matrix patterns", Proceedings of the 2011, International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19-20, 2011, Melbourne, Australia. Springer Berlin Heidelberg, **(2011)**, pp. 223-231.
[16] K. Muller, S. Mika, G. Ratsch and K. Tsuda, "An introduction to kernel-based learning algorithms", IEEE Trans Neural Netw, vol. 12, no. 2, **(2001)**, pp. 181-201.
[17] http://www.zjucadcg.cn/dengcai/Data/FaceData.html.
[18] S. Msji, A. Berg and J. Malik, "Efficient classification for additive kernel SVMs", IEEE Trans. Pattern Anal. Mach. Intell, vol. 1, no. 35, pp. 66-77, **(2013)**.
[19] A. Airola, T. Pahikkala, W. Waegeman, B. De Baets and T. Salakoski, "An experimental comparison of cross-validation techniques for estimating the area under the ROC curve", Comput.Stat. Data Anal, vol. 55, no. 4, **(2011)**, pp. 1828-1844.

