ICIC Express Letters
Part B: Applications
Volume **7**, Number **8**, August **2016**

ICIC International ©2016 ISSN 2185-2766
pp. 1657–1664

AN EFFICIENT NONLINEAR ONE-CLASS SVM BASED ON MATRIX PATTERNS

Yanyan Chen*, Yun Zhang and Ling Ling

College of Applied Science and Technology
Beijing Union University
No. 97, Beisihuan East Road, Chaoyang District, Beijing 100101, P. R. China
*Corresponding author: yykjtyanyan@buu.edu.cn

Received January 2016; accepted April 2016

Abstract. *In many fields, we often encounter one-class classification problems. The traditional vector-based one-class classification algorithms represented by one-class SVM (OCSVM) have limitations when matrix is considered as input data. This work addresses a nonlinear one-class classifier with matrix-based maximal margin classification paradigm. To this end, we formulate the Nonlinear One-class SVM Based on Matrix Patterns (NLMatOCSVM), which can directly use matrix as input. That helps to retain the data topology more efficiently in comparison with vector-based classifier. The efficiency of the proposed method is illustrated on two matrix-based human faces datasets. The experimental results indicate the validity of the new method.*
Keywords: One-class support vector machine, Matrix pattern, Kernel method, Nonlinear one-class classification problem

1. **Introduction.** 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. In one-class classification problems, usually, only one class is available, and the others are either expensive to acquire or difficult to characterize. One-class support vector machine proposed by Scholkopf et al. [1], is a popular one-class classifier. OCSVM seeks for a hyperplane in the maximal margin sense, which separates most of the samples from the origin. At present, there are lots of researches on one-class classification problems [2, 3, 4], all of which are based on vector space.

In practice, many of the original objects are represented as matrix, such as gray face image. The traditional vector-based one-class classification methods represented by OCSVM cannot work when matrix is considered. Although there are some methods converting matrix directly into vector, they may lead to structural information loss and data correlation damage [5]. All these reasons lead us to consider matrix representation and 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. [6] proposed a two-dimensional principal component analysis (2DPCA) which extracted features directly from the matrix. Chen et al. [7] developed a more general principal component analysis (MatPCA) to extract features based on matrix pattern. Wang and Chen [8] proposed a matrixized least squares support vector machine (MatLSSVM) that could directly classify the objects represented by matrix. Further, Wang et al. improved the MatLSSVM into an efficient kernelized classifier named kernel-based matrixized least square support vector machine (KMatLSSVM) [9]. Chen et al. derived a tensor-based one-class classification algorithm, named one-class support tensor machine (OCSTM) [10, 11] which took second order tensor (matrix) as input data. All the experiment results verified that the matrixized classifier had a superior classification performance to its vector version when it took matrix as input.

In this paper, we propose a nonlinear one-class classifier which directly takes matrix as input. We name it Nonlinear One-class SVM Based on Matrix Patterns. This new classifier helps to retain the data topology more efficiently in comparison with vector-based classifier. The efficiency of the proposed method is illustrated on two matrix-based human faces datasets. The experimental results indicate the validity of the new method.

The rest of this paper is organized as follows. In Section 2, we give a brief overview of related literature. The proposed NLMatOCSVM is described in Section 3 and the experimental evaluation is presented in Section 4. Finally, we provide some concluding remarks and suggestions for future work in Section 5.

2. Overview of Related Literature.

In this section, we first give a brief overview of standard OCSVM and the linear matrix-pattern-oriented one-class SVM (MatOCSVM) proposed by Yan et al. [12]. Then we introduce the kernel matrix invented by [13, 14].

2.1. OCSVM and MatOCSVM.

OCSVM 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 the target class, while all other instances are defined as outliers. Considering training data $\mathbf{x}_i \in \mathbb{R}^n$ $(i = 1, \ldots, l)$, the decision function relative to the membership of the sample $\mathbf{x}$ to the target class is given by: $f(\mathbf{x}) = ((\mathbf{w} \cdot \Phi(\mathbf{x})) - \rho) \geq 0$, where parameters $\mathbf{w}$ and ρ result from the modified optimization problem:

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

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

On the similar idea of OCSVM, Yan et al. [12] developed a new variant of OCSVM which directly took matrix as input. The decision function of MatOCSVM is formed as: $f(\mathbf{X}) = sgn\left(\mathbf{u}^T\mathbf{X}\mathbf{v} - \rho\right)$, where $\mathbf{X}$ is an $n_1 \times n_2$ matrix pattern, $\mathbf{u}$ is an n_1 dimensional left weight vector and $\mathbf{v}$ is an n_2 dimensional right weight vector.

The corresponding optimization problem of MatOCSVM is given by:

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

To solve the quadratic program problem (2), there are two procedures. Firstly, the right weight vector $\mathbf{v}$ is gotten by iteration. Secondly, in each iterative step k, an analytical solution to the left weight vector $\mathbf{u}$ can be gained with the fixed $\mathbf{v}$. The elaborate description about MatOCSVM can be found in [12].

2.2. The Kernel function for matrix.

Since MatOCSVM is a linear classifier operating on matrix, it cannot handle nonlinear classification problems. [13, 14] proposed a kernel function for matrix representation data, which used a nonlinear mapping function $\Phi(\mathbf{X}_i)$ to map $\mathbf{X}_i$ into a high dimensional feature space. The nonlinear mapping function for tensor $\mathbf{X}_i$ can be defined as:

$$\Phi\left(\mathbf{X}_i\right) = \begin{bmatrix} \varphi(z_{i1}) \\ \varphi(z_{i2}) \\ \vdots \\ \varphi(z_{in_1}) \end{bmatrix}, \tag{3}$$

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

$$K(\mathbf{X}_i, \mathbf{X}_j) = \Phi\left(\mathbf{X}_i\right)\Phi\left(\mathbf{X}_j\right)^T = \begin{bmatrix} \varphi(z_{i1})\varphi(z_{j1})^T & \dots & \varphi(z_{i1})\varphi(z_{jn_1})^T \\ \vdots & \ddots & \vdots \\ \varphi(z_{in_1})\varphi(z_{j1})^T & \dots & \varphi(z_{in_1})\varphi(z_{jn_1})^T \end{bmatrix}. \tag{4}$$

Specifically, we use RBF kernel function in this article, since it has been demonstrated that the RBF kernel usually outperforms the other kernels [15]. The ij-th element of the kernel matrix is:

$$\varphi(z_{ip_1})\varphi(z_{jp_2})^T = \mathrm{e}^{-\|z_{ip_1} - z_{jp_2}\|^2/2\sigma^2}, \tag{5}$$

and we call it the RBF kernel function for matrix, shortened as MatRBF kernel in the following passage.

3. **Nonlinear Metricized OCSVM.** Our nonlinear metricized OCSVM is fundamentally based on the same idea of MatOCSVM. In this section, we propose a new one-class classifier NLMatOCSVM based on matrix, which determines a decision function by mapping the matrix data into high dimensional feature space, so that the data points in the target class are separated by maximal margin from the origin.

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

As similar as MatOCSVM, the decision function of a nonlinear metricized OCSVM can be represented as follows:

$$f(\mathbf{X}) = sgn\left(\mathbf{u}^T\Phi\left(\mathbf{X}\right)\mathbf{v} - \rho\right), \quad \mathbf{u} \in \mathbb{R}^{n_1}, \quad \mathbf{v} \in \mathbb{R}^{n_2}, \tag{6}$$

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

Our NLMatOCSVM can be given by the following optimization problem:

$$\begin{aligned} \min_{\mathbf{u}\in\mathbb{R}^{n_1}, \mathbf{v}\in\mathbb{R}^{n_2}, \rho\in\mathbb{R}, \xi\in\mathbb{R}^l} \quad & \frac{1}{2}\|\mathbf{u}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i - \rho \\ \text{s.t.} \quad & \mathbf{u}^T\Phi(\mathbf{X}_i)\mathbf{v} \geq \rho - \xi_i \\ & \xi_i \geq 0, \quad i = 1, \dots, l \end{aligned} \tag{7}$$

We introduce positive Lagrange multipliers α_i, $\beta_i \geq 0$, $i = 1, \dots, l$, one for each of the inequality constrains. The Lagrangian function for problem (7) is

$$\begin{aligned} \mathcal{L}\left(\mathbf{u}, \mathbf{v}, \rho, \xi, \alpha, \beta\right) = & \frac{1}{2}\|\mathbf{u}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i - \rho \\ & - \sum_{i=1}^{l}\alpha_i\left(\mathbf{u}^T\Phi\left(\mathbf{X}_i\right)\mathbf{v} - \rho + \xi_i\right) - \sum_{i=1}^{l}\beta_i\xi_i. \end{aligned} \tag{8}$$

Then we can get the partial derivatives of $\mathcal{L}$ with respect to $\mathbf{u}$, $\mathbf{v}$, ρ, ξ_i,

$$\frac{\partial\mathcal{L}}{\partial\mathbf{u}} = \mathbf{u} - \sum_{i=1}^{l}\alpha_i\Phi(\mathbf{X}_i)\mathbf{v}, \tag{9}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = -\sum_{i=1}^{l} \alpha_i \Phi(\mathbf{X}_i)^T \mathbf{u}, \tag{10}$$

$$\frac{\partial \mathcal{L}}{\partial \rho} = \sum_{i=1}^{l} \alpha_i - 1, \tag{11}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = \frac{1}{\nu l} - \alpha_i - \beta_i. \tag{12}$$

From Equations (9) and (10), we see that $\mathbf{u}$ and $\mathbf{v}$ are dependent on each other, and cannot be solved independently. Hence, we resort to the iteration method for solving this optimization problem. The method can be described as follows.

First we fix $\mathbf{v}$. Let $\mu_1 = \|\mathbf{u}\|^2$ and $\mathbf{x}_i = \Phi(\mathbf{X}_i)\mathbf{v}$, and according to (7), we can construct the optimal quadratic programming problem to solve $\mathbf{u}$:

$$\begin{aligned} \min_{\mathbf{u},\xi,\rho} \quad & \frac{1}{2}\|\mathbf{u}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{l} \xi_i - \rho \\ \text{s.t.} \quad & (\mathbf{u} \cdot \mathbf{x}_i) \geq \rho - \xi_i \\ & \xi_i \geq 0, \quad i = 1, \ldots, l \end{aligned} \tag{13}$$

It can be seen that the optimization problem (13) is similar in structure to the standard OCSVM. To solve (13), we consider its Lagrangian function

$$\mathcal{L}(\mathbf{v}, \rho, \xi, \alpha, \beta) = \frac{1}{2}\|\mathbf{u}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{l} \xi_i - \rho - \sum_{i=1}^{l} \alpha_i \left(\mathbf{u}^T \mathbf{x}_i - \rho + \xi_i\right) - \sum_{i=1}^{l} \beta_i \xi_i. \tag{14}$$

According to Equations (9)-(12),

$$\begin{aligned} \mathcal{L}(\mathbf{v}, \rho, \xi, \alpha, \beta) &= -\frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j \mathbf{v}^T \Phi(\mathbf{X}_i) \Phi(\mathbf{X}_j)^T \mathbf{v} \\ &= -\frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j \mathbf{v}^T K(\mathbf{X}_i, \mathbf{X}_j) \mathbf{v}, \end{aligned} \tag{15}$$

where $K(\mathbf{X}_i, \mathbf{X}_j)$ is defined by Equation (4).

Thus we can get the dual problem of optimization problem (14):

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j \mathbf{v}^T K(\mathbf{X}_i, \mathbf{X}_j) \mathbf{v} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu l} \\ & \sum_{i=1}^{l} \alpha_i = 1, \quad i = 1, \ldots, l \end{aligned} \tag{16}$$

Solving (16) determines the lagrangian multipliers $\alpha_i{}^*$, then we can get

$$\mathbf{u} = \sum_{i=1}^{l} \alpha_i{}^* \Phi(\mathbf{X}_i) \mathbf{v}.$$

Then we can update $\mathbf{v}$ by gradient descent algorithm. According to Equation (10), we can obtain the iteration equation of $\mathbf{v}$:

$$\mathbf{v}_{t+1} = \mathbf{v}_t - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \mathbf{v}_t - \eta \left(-\sum_{i=1}^{l} \alpha_i \Phi(\mathbf{X}_i)^T \mathbf{u}\right), \tag{17}$$

where η is the learning rate and t is the iterative counter. According to the solution of $\mathbf{u}$, we have

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \eta \sum_{i,j=1}^{l} \alpha_i \alpha_j K(\mathbf{X}_i, \mathbf{X}_j) \mathbf{v}_t. \tag{18}$$

Thus, $\mathbf{u}$ and $\mathbf{v}$ can be obtained by iteratively solving the optimization problems (16) and (18). The optimal boundary is then determined by the support vector expansion:

$$f(\mathbf{X}) = sgn\left(\sum_{i=1}^{l} \alpha_i \mathbf{v}^T K(\mathbf{X}, \mathbf{X}_i) \mathbf{v} - \rho\right), \tag{19}$$

where the parameter ρ is calculated by:

$$\rho = mean_i \left(\sum_{i=1}^{l} \alpha_i \mathbf{v}^T K(\mathbf{X}_i, \mathbf{X}_i) \mathbf{v}\right). \tag{20}$$

On the whole, NLMatOCSVM is made up of two procedures. Firstly, the solution of right weight vector $\mathbf{v}$ can be gotten by gradient descent algorithm. Secondly, for every fixed $\mathbf{v}$, the optimization problem with left weight vector $\mathbf{u}$ can be solved by a kernel-based dual problem which is similar to the standard OCSVM.

4. **Experimental Evaluation.** In this section, we compare the performance of NLMatOCSVM with that of the standard OCSVM and linear MatOCSVM. After the statements about the datasets and preparation of experiments, we evaluate the proposed algorithms on matrix representation datasets.

4.1. **Preparation of experiments.** In this section, we focus on human face images datasets. There are two datasets we concern about: the ORL dataset [16] and the YALE dataset [17]. They are all grayscale images which can be represented as matrix. The ORL dataset consists of forty people's face images, everyone has ten different images and each image is 28×23 with 256 grayscale levels per pixel. The YALE dataset has fifteen people and eleven images for each person, and the images in YALE are the size of 100×100. And all the features in these two datasets are scaled to $[0, 1]$. Since we are interested in testing the effectiveness of proposed algorithms with matrix representation dataset, we do not perform cropping or resizing of the images which reduces the number of features in the data.

In all our simulations, we use RBF kernel $k(\mathbf{x}, \mathbf{y}) = \mathrm{e}^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$ in the standard OCSVM, for it has been demonstrated that the RBF kernel usually outperforms other kernels [15]. For the same reason, we use tensor kernel function $K(\mathbf{X}, \mathbf{Y})$ for NLMatOCSVM, which was defined in Section 2.

In one-class classification problems, the training samples all come from the target class, which means the samples in other class have nothing to contribute to the classifiers. Hence, we focus on the true positive rate (TPR) of the algorithms in all our simulations. The AUC, the area under the ROC curve, is always used to measure the performance of a one-class classifier [18]. In our experiments, we consider both TPR and AUC as the performance metrics for comparisons.

We use k-fold cross-validation on the training set to find the best parameters, while the value of k equals to the number of training samples. There are three tuning parameters: ν, σ and η. The possible choices for parameters are $\nu = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, $\sigma = \{2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 2^3, 2^4, 2^5\}$ and $\eta = \{10^{-1}, 10^0, 10^1\}$. 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.

4.2. **Experimental results and analysis.** For one class classification problems, we consider each person's faces as a target class, so there are 40 target classes in ORL and 15 target classes in YALE. And each target class has only 10 samples in ORL or 11 samples in YALE.

To verify the effectiveness of the algorithms above, 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.

Table 1. Averaged TPR on 40 target classes in ORL dataset

class	NLMatOCSVM	MatOCSVM	OCSVM	class	NLMatOCSVM	MatOCSVM	OCSVM
1	**0.675**	0.25	0.075	21	**0.725**	0.25	0.1
2	**0.675**	0.225	0.45	22	**0.7**	0.275	0.275
3	**0.85**	0.175	0.05	23	**0.75**	0.1	0.025
4	**0.775**	0.4	0.15	24	**0.675**	0.125	0.25
5	**0.675**	0.15	0.15	25	**0.675**	0.525	0.175
6	**0.725**	0.7	0.2	26	**0.85**	0.15	0.075
7	**0.725**	0.125	0.325	27	**0.8**	0.575	0.6
8	**0.725**	0.425	0.1	28	**0.8**	0.075	0.1
9	**0.75**	0.1	0	29	**0.85**	0.075	0.075
10	**0.7**	0.475	0.225	30	**0.825**	0.125	0.25
11	**0.75**	0.5	0.325	31	**0.8**	0	0.075
12	**0.825**	0.275	0.275	32	**0.7**	0.25	0.175
13	**0.75**	0.25	0.25	33	**0.825**	0	0.25
14	**0.85**	0.325	0.05	34	**0.875**	0.025	0.35
15	**0.7**	0.025	0.05	35	**0.825**	0.175	0.1
16	**0.45**	0.2	0.1	36	**0.775**	0.275	0.05
17	**0.875**	0.475	0.45	37	**0.625**	0.15	0.25
18	**0.8**	0.075	0.2	38	**0.825**	0.25	0.175
19	**0.825**	0.575	0.1	39	**0.825**	0.075	0
20	**0.7**	0.125	0.175	40	**0.8**	0.225	0.05

Table 2. Averaged TPR on 15 target classes in YALE dataset

class	NLMatOCSVM	MatOCSVM	OCSVM	class	NLMatOCSVM	MatOCSVM	OCSVM
1	**0.68**	0.36	0.38	9	**0.64**	0.24	0.36
2	**0.72**	0.24	0.36	10	**0.76**	0.14	0.34
3	**0.64**	0.02	0.4	11	**0.86**	0.2	0.42
4	**0.66**	0.04	0.46	12	**0.76**	0.1	0.5
5	**0.76**	0	0.56	13	**0.52**	0.08	0.64
6	**0.72**	0.08	0.72	14	**0.72**	0.2	0.46
7	**0.8**	0.08	0.26	15	**0.6**	0.14	0.42
8	**0.7**	0.1	0.12				

Obviously, it is an unbalanced classification problem, and it is more meaningful to concentrate on the true positive rate, the TPR. Table 1 and Table 2 summarize the TPR of each target class in the two datasets, and the best classification results are shown in boldfaces. We can see from the tables that the TPR of the NLMatOCSVM is outstanding in comparison with those of linear MatOCSVM or standard OCSVM.

We also calculate the AUC of the classifiers for evaluating the whole performance on the two datasets. In ORL dataset, the averaged AUC of 40 experiments of NLMatOCSVM is

0.88, which is not far from 0.77 of MatOCSVM and 0.96 of OCSVM. Similarly, in YALE dataset, the averaged AUC of 15 classes of NLMatOCSVM is 0.64, which is not far from 0.63 of MatOCSVM and 0.78 of OCSVM, either.

Thus, we can conclude that in these two matrix-based datasets, the TPR has been dramatically promoted by the metricized nonlinear classifier and the AUC among the classificers has no significant differences. That means the proposed metricized classifier NLMatOCSVM has greatly promoted the identification of the target samples in comparison with linear MatOCSVM or vector-based OCSVM, without losing the whole classification performance in the meanwhile.

5. **Conclusion and Future Work.** In this work we propose a new metricized nonlinear one-class classification algorithm. NLMatOCSVM uses matrix as input data, and aims to separate almost all the samples of target class from the origin with maximal margin. The benefit of the proposed algorithm is that the use of direct matrix is helpful to retain the data topology more efficiently. To solve the optimization problem corresponding to NLMatOCSVM, we use iteration method. In each iterative step, with a fixed right weight vector, the left weight vector can be estimated by solving a standard OCSVM optimization problem. And the right weight vector can be updated by the gradient descent algorithm. We validate our proposed algorithm on two matrix-based human faces datasets. As expected, the proposed algorithm yields better generalization performance.

However, there are some drawbacks of the proposed method. Since the iteration of solving parameters costs lots of time, the training time of NLMatOCSVM is much more than that of vector-based OCSVM. A possible direction on the work is to investigate more efficient computational methods to solve the optimization problems of NLMatOCSVM. Another interesting topic is to apply the NLMatOCSVM to real world classification, for the data point is originally expressed in matrix representation in many application areas.

Acknowledgment. The work is supported by 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] B. Scholkopf, J. C. Platt, J. S. 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] W. Zhu and P. Zhong, A new one-class SVM based on hidden information, *Knowledge-Based Systems*, vol.60, pp.35-43, 2014.
[3] 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, pp.4115-4121, 2011.
[4] Y. Chen, X. Zhou and T. S. Huang, One-class SVM for learning in image retrieval, *Proc. of IEEE Int'l Conf. on Image Processing*, Thessaloniki, Greece, 2001.
[5] D. Tao, X. Li, X. Wu and S. J. Maybank, Supervised tensor learning, *Knowledge and Information Systems*, vol.13, no.1, pp.1-42, 2007.
[6] 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, pp.131-137, 2004.
[7] S. Chen, Y. Zhu, D. Zhang and J. Yang, Feature extraction approaches based on matrix pattern: MatPCA and MatFLDA, *Pattern Recognition Letters*, vol.26, no.8, pp.1157-1167, 2005.
[8] Z. Wang and S. Chen, New least squares support vector machines based on matrix pattern, *Neural Processing Letters*, vol.26, pp.41-56, 2008.
[9] 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, pp.143-150, 2013.
[10] Y. Chen and P. Zhong, Linear one-class support tensor machine, *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol.9, no.9, 2016.

[11] Y. Chen, K. Wang and P. Zhong, One-class support tensor machine, *Knowledge-Based Systems*, vol.96, pp.14-28, 2016.

[12] Y. Yan, Q. Wang, G. Ni, Z. Pan and R. Kong, One-class support vector machines based on matrix patterns, *Proc. of the International Conference on Informatics, Cybernetics, and Computer Engineering*, Melbourne, Australia, pp.223-231, 2011.

[13] C. Gao and X. Wu, Kernel support tensor regression, *Procedia Engineering*, vol.29, pp.3986-3990, 2012.

[14] P. Daniusis and P. Vaitkus, Kernel regression on matrix patterns, *Lithuanian Mathematical Journal*, Spec. Edition, vol.48-49, pp.191-195, 2008.

[15] 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.

[16] AT&T Labs Cambridge, *The Olivetti and Oracle Research Laboratory Database of Faces*, http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html, 1994.

[17] *http://cvc.yale.edu/projects/yalefaces/yalefaces.html*.

[18] A. Airola, T. Pahikkala, W. Waegeman, B. D. 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, pp.1828-1844, 2011.