Neural Processing Letters (2007) 26:41–56
DOI 10.1007/s11063-007-9041-1

New Least Squares Support Vector Machines Based on Matrix Patterns

Zhe Wang · Songcan Chen

Received: 14 August 2005 / Accepted: 7 May 2007 / Published online: 13 June 2007

Abstract Support vector machine (SVM), as an effective method in classification problems, tries to find the optimal hyperplane that maximizes the margin between two classes and can be obtained by solving a constrained optimization criterion using quadratic programming (QP). This QP leads to higher computational cost. Least squares support vector machine (LS-SVM), as a variant of SVM, tries to avoid the above shortcoming and obtain an analytical solution directly from solving a set of linear equations instead of QP. Both SVM and LS-SVM operate directly on patterns represented by vector, i.e., before applying SVM or LS-SVM to a pattern, any non-vector pattern such as an image has to be first vectorized into a vector pattern by some techniques like concatenation. However, some implicit structural or local contextual information may be lost in this transformation. Moreover, as the dimension d of the weight vector in SVM or LS-SVM with the linear kernel is equal to the dimension $d_1 \times d_2$ of the original input pattern, as a result, the higher the dimension of a vector pattern is, the more space is needed for storing it. In this paper, inspired by the method of feature extraction directly based on matrix patterns and the advantages of LS-SVM, we propose a new classifier design method based on matrix patterns, called MatLSSVM, such that the new method can not only directly operate on original matrix patterns, but also efficiently reduce memory for the weight vector (d) from $d_1 \times d_2$ to $d_1 + d_2$. However like LS-SVM, MatLSSVM inherits LS-SVM's existence of unclassifiable regions when extended to multi-class problems. Thus with the fuzzy version of LS-SVM, a corresponding fuzzy version of MatLSSVM (MatFLSSVM) is further proposed to remove unclassifiable regions effectively for multi-class problems. Experimental results on some benchmark datasets show that the proposed method is competitive in classification performance compared to LS-SVM, fuzzy LS-SVM (FLS-SVM), more-recent MatPCA and MatFLDA. In addition, more importantly, the idea used here has a possibility of providing a novel way of constructing learning model.

Z. Wang · S. Chen (✉)
Department of Computer Science & Engineering, Nanjing University of Aeronautics & Astronautics, 29 Yudao Street, Nanjing, Jiangsu 210016, China
e-mail: s.chen@nuaa.edu.cn

Keywords Support vector machine (SVM) · Least squares support vector machine (LS-SVM) · Fuzzy least squares support vector machine (FLS-SVM) · Vector pattern · Matrix pattern · Pattern recognition

1 Introduction

Support vector machine (SVM), based on the statistical learning theory, has effectively been developed [1–3] and applied successfully in machine learning. The goal of SVM aims to minimize the Vapnik-Chervonenkis (VC) dimension by finding the optimal hyperplane with the maximal margin, where the margin is defined as the distance of the closest point, in each class, to the separating hyperplane. And it has two key advantages: a general purpose linear learning algorithm and a problem specific kernel that computes the inner product of input data points in a feature space. However, finding this optimal hyperplane by solving a constrained optimization criterion using quadratic programming (QP) leads to higher computational cost. Least squares support vector machine (LS-SVM) established by Suykens and Vandewalle [4,5] can obtain an analytical solution directly from solving a set of linear equations instead of QP through replacing inequality constraints with equality constraints in formulation of the conventional SVM. This can efficiently reduce the computational cost and at the same time still makes LS-SVM inherit the aforementioned two advantages of the conventional SVM.

Generally, a pattern or data object that both SVM and LS-SVM deal with is a vector. Therefore, when a pattern to be processed is an image itself, for instance, a face, the image first has to be transformed or vectorized into a vector pattern by concatenating its pixels in some way. This type of vector representation is natural in most of data analyses [6]. However, one inherent problem of the image-as-vector representation lies in that the spatial redundancies within each image matrix are not fully utilized, and some of the information about local spatial relationships is lost [7,8]. On the other hand, although computational efficiency in LS-SVM is raised compared to SVM, loss of sparseness in its solution [9] makes every input pattern (e.g. the dimension of an input pattern is $d_1 \times d_2$ and the number of the input patterns is l) contribute to the model, thus leading to LS-SVM have to store the training set for subsequent classification such that the memory overhead is greatly increased. In fact, for LS-SVM with linear kernel to be mainly discussed here, we can save so-needed storage space directly by storing the $d_1 \times d_2$-dimensional weight vector and the bias rather than the whole training set. However, even in such a case, the storage space requirement will still be rather large for large dimensionality patterns such as an image due to the dimension of the weight vector is equal to that of the input pattern. Therefore, intuitively, directly manipulating original matrix patterns by means of the matrix algebra method seems simpler. Several researchers have made such attempts along this line. Yang et al. [10] proposed two-dimensional principal component analysis (2DPCA) that extracts features directly from the 2D images. Through the experiments on several well-known benchmark faces, this method is shown to be better than classical PCA in favor of both image classification and reduction of computation complexity for feature extraction. Ye et al. [11] and Li and Yuan [12] also respectively proposed two-dimensional linear discriminant analysis (2DLDA) based on image matrix, which overcomes the singularity problem implicitly in LDA and achieves competitive recognition accuracy on face identification. But all the 2D methods above are only used to deal with two-dimensional image patterns themselves. Chen et al. [7] went further and developed a more general method, called MatPCA and MatFLDA, to extract features based on matrix patterns reshaped from an original one-dimensional or 2D pattern. Compared with the conventional methods such as PCA and LDA of extracting features based on vector patterns, Chen's method first matrixizes a one-dimensional or 2D pattern into a corresponding matrix pattern before exacting features. In doing so, the information, generally, should not be lost due to that such newly-formed matrix pattern still retains all its feature components, more likely, some new implicit structural or contextual information can additionally be introduced. Inspired by both Chen's matrixization method and the analytically-solvable advantages of LS-SVM, in this paper, we introduce similar matrixization method into LS-SVM and expect that this introduction can combine the advantages of both the matricization method and LS-SVM and finally develop a new variant of LS-SVM directly operating on matricized patterns (called MatLSSVM, for short). Compared with LS-SVM with linear kernel, which has a $d(= d_1 \times d_2)$-dimensional weight vector w, MatLSSVM has the two weight vectors u with d_1 dimensionality and v with d_2 dimensionality that respectively act on the left and right sides of the matrix pattern A with $d_1 \times d_2$ dimensionality so that the matrix pattern can be directly manipulated. Through this transformation, the information of the matrix pattern may be effectively utilized and the space which is used to store weight vectors is reduced from $d_1 \times d_2$ for LS-SVM with linear kernel to $d_1 + d_2$ for MatLSSVM. However, LS-SVM is formulated only for two-class problems and unclassifiable regions exist when extended to multi-class problems. Similarly, MatLSSVM also has the same problem. In order to remove such unclassifiable regions, Tsujinishi and Abe [13] proposed fuzzy version (FLS-SVM) of LS-SVM for multi-class problems. Borrowing their idea, similar fuzzy version MatFLSSVM is also presented here for attacking the same unclassifiable region problem existed in MatLSSVM.

We compare MatLSSVM and MatFLSSVM with corresponding LS-SVM, FLS-SVM, MatPCA and MatFLDA on some benchmark machine learning datasets. Through the experiments, we find that the proposed matrixization approaches are competitive with their corresponding vector versions.

The rest of this paper is organized as follows. In Sect. 2, LS-SVM and FLS-SVM are reviewed respectively. We describe MatLSSVM and MatFLSSVM in detail in Sect. 3. In Sect. 4 we show the performance comparison among these algorithms. Section 5 further gives the relationship between LS-SVM and MatLSSVM. Finally, we conclude in Sect. 6.

2 An Overview of LS-SVM and FLS-SVM

LS-SVM proposed by Suykens and Vandewalle [4,5] is a least squares version of SVM. Due to the equality type constraints instead of inequality constraint, the solution follows from solving a set of linear equations instead of QP and the computational cost of LS-SVM is rather lower than that of SVM. However, LS-SVM is formulated for two-class problems and unclassifiable regions exist when extended to multi-class problems. So FLS-SVM [13] was proposed as a fuzzy version of LS-SVM to effectively resolve unclassifiable regions for multi-class problems. In this section we will respectively review LS-SVM and FLS-SVM.

2.1 LS-SVM

Let $S = \{(x_i, y_i)|i = 1, \dots, l\} \subset R^n \times \{+1, -1\}$ be a set of examples. The decision function that we want to achieve in LS-SVM is given by:

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b, \tag{1}$$

where α_i are real constants and b is a bias. The output of $f(x)$ is 1 if its value is greater than 0, -1 otherwise. In (1), $K(x, x_i)$ is a kernel function, and α_i and b are unknown and can be solved through the primal problem (P) defined below:

$$\min_{w,b,\xi} \quad \frac{1}{2}w^t w + \frac{C}{2}\sum_{i=1}^{l}\xi_i^2, \tag{2}$$

subject to the equality constraints:

$$y_i(w^t\Phi(x_i)+b) = 1-\xi_i, \quad i=1,\ldots,l, \tag{3}$$

where $\Phi(.)$ is a linear or nonlinear function which maps the input space into a higher dimensional feature space, w is a weight vector to be determined, C is a regularization constant and ξ_is are slack variables. Therefore, we construct the Lagrangian:

$$L(w,b,\xi,\alpha) = \frac{1}{2}w^t w + \frac{C}{2}\sum_{i=1}^{l}\xi_i^2 - \sum_{i-1}^{l}\alpha_i\left[y_i(w^t\Phi(x_i)+b)-1+\xi_i\right], \tag{4}$$

where α_i are Lagrange multipliers (i.e. α_i in (1)). The necessary conditions for the optimality are:

$$\begin{cases} \dfrac{\partial L}{\partial w} = 0 \Rightarrow w = \sum\limits_{i=1}^{l}\alpha_i y_i \Phi(x_i), \\ \dfrac{\partial L}{\partial b} = 0 \Rightarrow \sum\limits_{i=1}^{l}\alpha_i y_i = 0, \\ \dfrac{\partial L}{\partial \xi_i} = 0 \Rightarrow \alpha_i = C\xi_i, \quad i=1,\ldots,l, \\ \dfrac{\partial L}{\partial \alpha_i} = 0 \Rightarrow y_i(w^t\Phi(x_i)+b)-1+\xi_i = 0, \quad i=1,\ldots,l. \end{cases} \tag{5}$$

By elimination of w and ξ_i, the solution is given by

$$\begin{bmatrix} 0 & -Y^t \\ Y & \Omega + C^{-1}I \end{bmatrix}\begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{1} \end{bmatrix}, \tag{6}$$

where $Y = [y_1,\ldots,y_l]^t$, $\Omega_{ij} = y_i y_j \Phi(x_i)^t\Phi(x_j)$, $\vec{1} = [1,\ldots,1]^t$. The function $\Phi(x)$ in (6) is related to the kernel function $K(x, x_i)$ by imposing

$$\Phi(x_i)^t\Phi(x_j) = K(x_i, x_j), \tag{7}$$

which is motivated by Mercer's Theorem.

2.2 FLS-SVM

As a fuzzy version of LS-SVM, FLS-SVM introduces fuzzy membership functions to resolve unclassifiable regions for multi-class problems in conventional pairwise classification [14]. In conventional pairwise classification for multi-class problem [14], there is a binary classifier for each possible pair of classes and the number of the total pairs is $k(k-1)/2$ for k-class problem. The decision function for the pair of classes i and j is given by:

$$f_{ij}(x) = w_{ij}^t\Phi(x) + b_{ij}, \quad i,j = 1\ldots k, \tag{8}$$

where x is a certain vector pattern and $f_{ij}(x) = -f_{ji}(x)$. Then for the pattern x we calculate

$$f_i(x) = \sum_{i=1, i \neq j}^{k} \mathrm{sign}(f_{ij}(x)), \tag{9}$$

where $\mathrm{sign}(f(x)) = \begin{cases} 1 & f(x) > 0, \\ 0 & \text{otherwise,} \end{cases}$ and this pattern is classified into the class j according to

$$j = \arg\max_{i=1,\ldots,k} f_i(x). \tag{10}$$

But if Eq. 10 is satisfied for plural js, x is unclassifiable [13]. Thus all these pattern xs form unclassifiable regions.

To avoid this, Tsujinishi and Abe [13] introduced a fuzzy membership function and defined the one-dimensional membership function $m_{ij}(x)$ as follows:

$$m_{ij}(x) = \begin{cases} 1 & \text{for } f_{ij}(x) \geq 1, \\ \\ f_{ij}(x) & \text{otherwise.} \end{cases} \tag{11}$$

In [13], using two operators, the minimum and the average operators, determines the membership to which class the unknown pattern x belongs. Here we only adopt the minimum operator due to that it can more effectively resolve the unclassifiable region problem.

With the minimum operator, the final membership $m_i(x)$ of the x for class i can be determined by

$$m_i(x) = \min_{j=1,\ldots,k} m_{ij}(x), \tag{12}$$

Consequently, the x is classified into the class j where

$$j = \arg\max_{i=1,\ldots,k} m_i(x). \tag{13}$$

3 MatLSSVM and MatFLSSVM

Due to the advantages of both the matrixization method and LS-SVM, we here propose the matrixizied version of LS-SVM, MatLSSVM, for two-class problems operating directly on matricized patterns. To remove unclassifiable regions effectively for multi-class problems, a corresponding fuzzy version of MatLSSVM (MatFLSSVM) is also further proposed. In the matricized LS-SVMs, the decision functions for MatLSSVM and MatFLSSVM are both taken as the following form:

$$f(A) = u^t A v + b, \tag{14}$$

where A is a d_1-by-d_2 matrix pattern, u is a d_1-dimensional (left) weight vector, v is a d_2-dimensioanal (right) weight vector and b is a bias as in LS-SVM. Obviously, for the same pattern A, the space that the matricized LS-SVMs need to store the weight vectors is $d_1 + d_2$ and the space that the vector counterparts with linear kernel need to store the weight vector is $d_1 \times d_2$. The ratio of the spaces that the two versions need is $(d_1 + d_2)/(d_1 \times d_2)$. Table 1 shows that for matrix patterns with different size, the higher dimension the pattern has, the less memory new model needs to store weight vectors compared with LS-SVM.

Table 1 Spaces needed for storing weight vectors of MatLSSVM and LS-SVM with linear kernel respectively based on matrix pattern and vector pattern for ORL datasets[a] with different matrix size (one unit space/one image)

Image size	$d_1 + d_2$ (Matrix)	$d_1 \times d_2$ (Vector)	$(d_1 + d_2)/(d_1 \times d_2)$
14×11	25	154	1:6.16
28×23	51	644	1:12.63
56×46	102	2576	1:25.25
112×92	204	10304	1:50.51

[a] Available at http://www.uk,research.att.com/facedatabase.html

3.1 MatLSSVM Construction

Firstly, we consider the two-class classification problem in the matrix case. Let $S = \{(A_i, y_i)|i = 1, \ldots, l\} \subset R^{d_1 \times d_2} \times \{+1, -1\}$ be a set of examples. MatLSSVM aims at constructing a classifier with the decision function of the form (14). And we require that for the patterns in S, the following condition (15) must be met to the greatest degree:

$$f(A_i) = u^t A_i v + b \begin{cases} \geq 1, & \text{if } y_i = 1; \\ \leq -1, & \text{if } y_i = -1; \end{cases} \quad i = 1, \ldots, l. \tag{15}$$

Integrating their corresponding binary class labels, we thus have

$$y_i f(A_i) = y_i(u^t A_i v + b) \geq 1, \quad i = 1, \ldots, l. \tag{16}$$

In order to deal with linearly inseparable set of the examples, as in LS-SVM, we introduce a set of slack variables ξ_i into Eq. 16 as follows:

$$y_i f(A_i) = y_i(u^t A_i v + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l. \tag{17}$$

Considering that A is a d_1-by-d_2 matrix, thus Av is a d_1-dimensional vector and as a result, the original set of matrix examples S can be transformed into a set of new low-dimensional vector examples $S_{vec} = \{(A_i v, y_i)|i = 1, \ldots, l\} \subset R^{d_1 \times 1} \times \{+1, -1\}$ for each fixed v. Then, on the S_{vec}, similarly to LS-SVM, the primal problem (P) for MatLSSVM can be formulated below:

$$\min_{u,v,b,\xi} \quad \frac{1}{2} u^t u + \frac{C}{2} \sum_{i=1}^{l} \xi_i^2, \tag{18}$$

subject to the equality constraints:

$$y_i(u^t A_i v + b) = 1 - \xi_i, \quad i = 1, \ldots, l, \tag{19}$$

where (19) is quadratic due to that both u and v are unknown. Through (18) and (19), we can construct the Lagrangian:

$$L(u, v, b, \xi, \alpha) = \frac{1}{2} u^t u + \frac{C}{2} \sum_{i=1}^{l} \xi_i^2 - \sum_{i=1}^{l} \alpha_i \left[y_i(u^t A_i v) + b) - 1 + \xi_i \right], \tag{20}$$

where α_i are Lagrange multipliers. Then we get the partial derivatives of L respectively with respect to u, b,ξ_i, α_i and set them equal to zero:

$$\frac{\partial L}{\partial u} = u - \sum_{i=1}^{l} \alpha_i y_i A_i v = 0, \tag{21}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{l} \alpha_i y_i = 0, \tag{22}$$

$$\frac{\partial L}{\partial \xi_i} = C\xi_i - \alpha_i = 0, i = 1, \ldots, l, \tag{23}$$

$$\frac{\partial L}{\partial \alpha_i} = -\{y_i(u^t A_i v + b) - 1 + \xi_i\} = 0, i = 1, \ldots, l. \tag{24}$$

According to (21), we find that u and v are interdependent. Thus it is impossible to analytically compute the optimal u and v simultaneously. However, for a fixed v, from (21) to (24), we can observe that the equations are formally the same as those in (5). Thus, the weight u can be solved by using the same approach as that of LS-SVM. On the other hand, in order to make (18) minimal, according to optimization theory, we develop a gradient descent algorithm to derive the solution to the v by (25):

$$\frac{\partial L}{\partial v} = -\sum_{i=1}^{l} \alpha_i y_i A_i^t u. \tag{25}$$

Details are given below.

We first substitute the fixed v into (21) and (24). By elimination of u and ξ_i, the similar solution to that of LS-SVM can be gotten by

$$\begin{bmatrix} 0 & -Y^t \\ Y & \Omega + C^{-1}I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{1} \end{bmatrix}, \tag{26}$$

where $Y = [y_1, \ldots, y_l]^t$, $\vec{1} = [1_1, \ldots, 1_l]^t$, $\Omega_{ij} = y_i y_j (A_i v)^t (A_j v) = y_i y_j v^t A_i^t A_j v$. For Ω_{ij}, we can rewrite it by $\Omega_{ij} = y_i y_j v^t K(A_i, A_j) v$, where $K(A_i, A_j)$ is the kernel *matrix* that operates on the matrix pair A_i, A_j, and defined as $K(A_i, A_j) = A_i^t A_j$. Further, u can be gotten by (21) and v is updated by (25), i.e.

$$v_{t+1} = v_t - \eta \frac{\partial L}{\partial v_t} = v_t - \eta \left(-\sum_{i=1}^{l} \alpha_i y_i A_i^t u_t \right), \tag{27}$$

where η is the learning rate and t is the iterative counter. The full algorithm of MatLSSVM is summaried as follows:

```
1. Initialize $v_0$, $\eta$, $maxIter$, $\varepsilon$ and $C$;
2. Compute the vector pattern space $S_t = \{x_1, \ldots x_i, \ldots x_l\} = \{A_1 v_t, \ldots A_i v_t, \ldots A_l v_t\}$
   from the matrix pattern space $S = \{A_1, \ldots A_i, \ldots A_l\}$.
3. Get $b_t$, $\alpha_i$ through (26), then $u_t = \sum_{i=1}^{l} \alpha_i y_i A_i v_t$;
4. Update $v$ through (27), i.e. $v_{t+1} = v_t - \eta \frac{\partial L}{\partial v_t} = v_t - \eta \left( -\sum_{i=1}^{l} \alpha_i y_i A_i^t u_t \right)$;
5. If $\|v_{t+1} - v_t\| > \varepsilon$ and $t < maxIter$ then $t = t + 1$, goto 2; else stop.
```

where v_0 is the initial value of v, *maxIter* is the maximal iterative count and ε is a termination condition. Consequently, we can represent the decision function of the classifier for input matrix pattern Z

$$f(Z) = u^t Z v + b \begin{cases} > 0 & then\ Z \in class + 1 \\ < 0 & then\ Z \in class - 1 \end{cases}, \tag{28}$$

where the two weight vectors u and v, the bias b are obtained in the process of training.

On the whole, MatLSSVM is made up of two procedures. Firstly, the final solution to v can be gotten by an iterative algorithm. Secondly, for every fixed v, an analytical solution to u can be gained similarly to LS-SVM. Here, for a fixed v, the solution to u is free of local minima due to the solution to a convex optimization [15] and at the same time, the computational cost may obviously be reduced because of the dimension reduction of patterns. However, in solving v, its optimality may not be guaranteed due to non-convexity of the criterion (18) for v. In addition, it is worth noting that if we want the problem (18) to be convex, we may alternatively take only u instead of both u and v as unknown and view the vector v in such a case as a hyperparameter at the kernel level of (26).

3.2 MatFLSSVM

Similarly to FLS-SVM, MatFLSSVM as a fuzzy version of MatLSSVM is used to solve the unclassifiable regions for multi-class problems. In MatFLSSVM, the decision function for the pair of classes i and j of a k-class problem is given by:

$$f_{ij}(A) = u_{ij}^t A v_{ij} + b_{ij}, \quad i, j = 1 \ldots k, \tag{29}$$

where A is defined as before and the pair decision function f_{ij} satisfies $f_{ij}(A) = -f_{ji}(A)$ for all i and j.

The one-dimensional membership function $m_{ij}(A)$ can be similarly defined as:

$$m_{ij}(A) = \begin{cases} 1 & \text{for} \quad f_{ij}(A) \geq 1, \\ f_{ij}(A) & \text{otherwise.} \end{cases} \tag{30}$$

With the minimum operator, the final membership $m_i(A)$ of the x for class i can be determined by

$$m_i(A) = \min_{j=1,\ldots,k} m_{ij}(A), \tag{31}$$

Consequently, the A is classified to the class j making

$$j = \arg\max_{i=1,\ldots,k} m_i(A). \tag{32}$$

4 Experiments

In this section we will experimentally compare the classification performance among LS-SVM, FLS-SVM, MatLSSVM and MatFLSSVM, and at the same time, also give the comparison results among MatPCA [7], MatFLDA [7], MatLSSVM and MatFLSSVM.

4.1 Description of Experimental Datasets

These experiments are conducted on the benchmark datasets including both those in a matrix representation, for example, the ORL face database and the Letter text-base,[1] and those in a vector representation: Iris (4 attributes/ 3 classes/150 data)[2], Waveform (21/3/1500)[2], Wine dataset (12/3/178) that is generated through the last one attribute of the original Wine dataset with 13 attributes[2] being omitted (by us) mainly for producing more assembling matrix patterns, Water-treatment (denoted Water-T.) (38/2/116),[2] Sonar (60/2/208),[3] Musk Clean2 (denoted M.C.2) (166/2/6598)[2] and M.C.2 (160/2/6598) that is the same dataset in M.C.2 with 166 attributes with the last six dimensions of each pattern being also omitted (by us) mainly for generating more assembling matrix patterns. ORL faces base contains 400 grey human face images of 40 persons, 10 different images each person and each image size is normalized to 28×23. Letter dataset contains 10 text classes consisted of digits 0–9 with each class having 50 samples and each sample is a 24×18 matrix pattern.

4.2 Set-up of experiments

In our experiments, each dataset is randomly divided into the two no-overlapping parts with the one for training and the other one for testing. Then, for each such classification problem, 10 independent runs are performed and their classification accuracies on the test sets are averaged and reported as the way in [7]. Here, the involved parameters include the maximal iterative count *maxIter*, ε, v_0, the learning rate η, and the regularization constant C. We initialize *maxIter*=5000, $\varepsilon = 10^{-3}$. In general, v_0 can be initialized with an arbitrary vector, but for simplicity and fairness of comparison among the different algorithms, we definitely initialize $v_0 = [1, \ldots, 1, \ldots, 1]^t$. The learning rate η is selected from the set {1, 10, 100, 1000} and the range of the regularization constant C is from 2^{-6} to 2^{10} with each step by multiplying 2. Consequently, the optimal η and C for each dataset, correspond to the best average test accuracy based on the ten independent runs. At the same time, the kernel functions between LS-SVM and FLS-SVM both choose the linear kernel, i.e., $K(x, x_i) = x^t x_i$ with respect to $\Phi(x) = x$. Correspondingly,$K(A_i, A_j)$ (*matrix*) of (26) in MatLSSVM and MatFLSSVM is set to $A_i^t A_j$.

It is worth pointing out that although all the patterns in both ORL and Letter datasets are 2D matrix themselves and the proposed methods can directly use them as the input patterns, we can still reshape them to another corresponding matrix patterns with different size as done in the experiments below. As for original vector patterns, we can matrixize them to a corresponding matrix patterns by some concatenation or assembling techniques.

4.3 Experimental Results

4.3.1 Comparison Among LS-SVM, FLS-SVM, MatLSSVM and MatFLSSVM

Table 2 demonstrates all classification accuracies on different datasets under the above experimental conditions. The best classification accuracy for each dataset is shown in boldfaces. First of all, we examine matrixization effect by only comparing the classification performance of MatLSSVM with LS-SVM. Here, both classifiers use the conventional pairwise

[1] Available at http://sun16.cecs.missouri.edu/pgader/CECS477/NNdigits.zip

[2] Available at http://www.ics.uci.edu/~mlearn/MLRository.html

[3] Available at ftp://ftp.cs.cmu.edu/afs/cs/project/connect/bench/

Table 2 Classification accuracies comparison among LS-SVM, FLS-SVM, MatLSSVM and MatFLSSVM

Datasets (attributes;classes)	Classifiers			
	LS-SVM (%)	MatLSSVM (%)	FLS-SVM (%)	MatFLSSVM (%)
Sonar (60A;2)	75.74*($C=2$)	**75.83($C=2^{-2}$;2×30)**	/	/
		71.48($C=2^{2}$;30×2)		
		74.35($C=2^{-1}$;3×20)		
		70.74($C=2^{-1}$;20×3)		
		69.81($C=2^{0}$;10×6)		
		75.46($C=2^{-1}$;6×10)		
Water-T. (38A;2)	94.85($C=2^{-6}$)	96.21($C=2^{8}$;19×2)	/	/
		98.33($C=2^{2}$;2×19)		
M.C.2 (166A;2)	**87.78*($C=2^{4}$)**	87.22($C=2^{6}$;83×2)	/	/
		86.85($C=2^{-5}$;2×83)		
M.C.2 (160A;2)	**88.31*($C=2^{-2}$)**	88.29($C=2^{-4}$;80×2)	/	/
		87.00($C=2^{-5}$;2×80)		
		84.47($C=2^{-4}$;16×10)		
		82.31($C=2^{0}$;10×16)		
ORL (28×23;40)	96.40($C=2^{5}$)	95.30($C=2^{5}$;28×23)	**96.50*($C=2^{5}$)**	95.35($C=2^{5}$;28×23)
		94.20($C=2^{-1}$;23×28)		94.70($C=2^{-1}$;23×28)
		93.50($C=2^{-5}$;161×4)		94.05($C=2^{-5}$;161×4)
		92.20($C=2^{-6}$;4×161)		92.40($C=2^{-6}$;4×161)
		94.20($C=2^{-1}$;46×14)		94.95($C=2^{-1}$;46×14)
		95.00($C=2^{-2}$;14×46)		95.30($C=2^{-2}$;14×46)

Table 2 continued

Datasets (attributes;classes)	Classifiers			
	LS-SVM (%)	MatLSSVM (%)	FLS-SVM (%)	MatFLSSVM (%)
Letter (24×18;10)	92.28 (C = 2)	92.55(C = 2^{-3};2×216)	92.36* (C = 2)	**93.10(C = $\mathbf{2^{-3}}$;2 x 216)**
		91.85(C = 2^{-4};216×2)		92.55(C = 2^{-4};216×2)
		91.40(C = 2^{-3};4×108)		92.15(C = 2^{-3};4×108)
		88.75(C = 2^{-3};108×4)		89.65(C = 2^{-3};108×4)
		89.40(C = 2^{-6};24×18)		89.90(C = 2^{-6};24×18)
		86.70(C = 2^{-6};18×24)		88.20(C = 2^{-6};18×24)
Iris (4A,3)	97.47(C = 2^0)	**97.60(C = $\mathbf{2^0}$;2 x 2)**	97.47* (C = 2^0)	**97.60(C = $\mathbf{2^0}$;2 x 2)**
Waveform	86.01(C = 2^4)	86.05(C = 2^{-1};3×7)	86.06*	**86.23(C = $\mathbf{2^{-1}}$;3 x 7)**
(21A;3)		85.20(C = 2;7×3)	(C = 2^4)	85.57(C = 2;7×3)
Wine (12A;3)	**96.98(C = $\mathbf{2^{-1}}$)**	94.26(C = 2^2;6×2)	**96.98(C = $\mathbf{2^{-1}}$)**	94.72(C = 2^2;6×2)
		90.00(C = 2^{-1};2×6)		90.28(C = 2^{-1};2×6)
		89.25(C = 2^{-2};4×3)		91.32(C = 2^{-2};4×3)
		90.00(C = 2^{-2};3×4)		90.19(C = 2^{-2};3×4)

The asterisk '*' denotes that the difference between the matrixization and vector approaches is not significant at 5% significance level, i.e., t-value < 1.7341

All the results are gotten in the best regularization constant C (C = 2^{-6}–2^{10}) and corresponding matrix size in the bracket. The best result for each dataset is denoted in boldface strategy to deal with those multi-class datasets: ORL, Letter, Iris, Waveform and Wine. From this table, we find that for image patterns themselves such as ORL and Letter datasets, MatLSSVM and LS-SVM exhibit their separate advantage. For ORL dataset, the classification accuracy of MatLSSVM on the original 28×23 matrix pattern outperforms that on all other reshaped matrix patterns but is decreased by 1.1% compared with LS-SVM. Conversely, on Letter datasets, the results are exactly opposite. For MatLSSVM, the best accuracy is not achieved on the original 24×18 matrix pattern but on the 2×216 one. Moreover, compared with LS-SVM, MatLSSVM accuracy is increased by 0.27%. From these facts, we believe whether the matrixization method obtains beneficial structural information for image pattern classification or not depends on different matrixiziation for the same dataset and different datasets. For the vector patterns from the rest seven datasets used here, similarly to image patterns, the performance raise of MatLSSVM over LS-SVM does not always hold. In fact, on the four datasets (Sonar, Water-T., Iris, Waveform), the matrixization method improves performance from slight to distinct under at least one matixizing pattern compared with LS-SVM and whereas on another three datasets (M.C.2(166A), M.C.2(160A), Wine), the results are opposite. On the whole, the matrixization method improves performance on four out of seven datasets and especially distinct on Water-T. (achieving about 4%). On the other hand, on those datasets that their performances are degraded, the matrixization method achieves about 2.5% decrease on Wine in the worst case but only slight on the rest. Therefore, similarly to the image pattern datasets, for the vector pattern datasets, whether MatLSSVM can really increase performance or not depends on different matrixization even for the same dataset and different datasets. From different performance exhibitions for different matrixizing patterns on the same datasets, we are reminded that the matrixization also has its own two sides: one side is for it to indeed facilitate representing some structural information if some matrixizing pattern is coincidental to nature of those data such as Letter and Water-T. datasets and the other is opposite, i.e., it exists some possibility of breaking down the structure of data itself, especially for 1D vector pattern such as Wine datasets. In short, matrixization for vector patterns just gives us one more option in a lot of classification methods.

In order to get better accuracies on multi-class problems, both FLS-SVM and MatFLSSVM are also implemented. The results are tabulated in Table 2 from which we can clearly observe that on these multi-class datasets, both FLS-SVM and MatFLSSVM can consistently be better or comparable to in classification accuracy LS-SVM and MatLSSVM, respectively. These experimental results accord with the theoretical analysis in [13] and this is the reason why we further propose MatFLSSVM. Moreover, on these multi-class datasets, the similar analyses for experimental results between MatFLSSVM and FLS-SVM can also be obtained and seem to follow the same trend between MatLSSVM and LS-SVM.

In order to further demonstrate that the proposed matrixization approaches (MatLSSVM and MatFLSSVM) are competitive with their corresponding vector versions (LS-SVM and FLS-SVM), we perform the t-test on the classification results of the 10 runs to calculate the statistical significance of the proposed matrixization approaches. The null hypothesis H_0 demonstrates that there is no significant difference between the mean number of patterns correctly classified by the proposed matrixization approaches and their corresponding vector versions. If the hypothesis H_0 of each dataset is accepted at the 5% significance level, i.e., the t-test value is less than 1.7341, the corresponding results in Table 2 will be denoted by the asterisk '*'. Consequently, from Table 2, it can be clearly found that the proposed matrixization approaches possess comparable classification performance with their corresponding vector approaches. On the whole, compared with the vector approaches, the main advantage of the proposed matrixization versions is less space demanding but at the price of an iterative implementation.

4.3.2 Comparison Among MatPCA, MatFLDA, MatLSSVM, MatFLSSVM

In this section we also make a compare in classification performances among MatLSSVM and MatFLSSVM, and other matrixization methods such as MatPCA and MatFLDA. MatPCA and MatFLDA are matricized versions with respect to PCA and FLDA respectively and are used to extract features on which the Nearest Neighbor (NN) is often used as the classification method. Table 3 demonstrates all the best classification accuracies of all the above methods and the best result for each dataset is denoted in boldface. According to the table, the best accuracies of MatLSSVM and MatFLSSVM are all better than those of MatPCA and MatFLDA on all datasets except Sonar dataset. In particular on Letter, Waveform, Wine, M.C.2(160A) datasets, their accuracies are both increased by more than 5%, respectively. Further, due to use of the NN classification in MatPCA and MatFLDA, every training pattern in dataset has to be stored and thus leading to rather large memory space. In contrast, both MatLSSVM and MatFLSSVM only need store their weight vectors and biases, producing great saving for the storing space.

5 Further Discussion on the Relationship between LS-SVM and MatLSSVM

In this section, we will further reveal the relationship between LS-SVM and MatLSSVM from the point of view of both theory and experiment for better understanding.

The decision function of LS-SVM with linear kernel (i.e. $\Phi(x) = x$) is defined as follows:

$$f(x) = w^t x + b, \tag{33}$$

where x is a $(d_1 \times d_2)$-dimensional vector pattern, then dimension of the weight vector w is $d_1 d_2 \times 1$ and b is a bias.

The decision function of MatLSSVM is (14):

$$f(A) = u^t A v + b,$$

where A is a d_1-by-d_2 matrix pattern, u is a d_1-dimensional (left) weight vector, v is a d_2-dimensioanal (right) weight vector and b is a bias.

Then, we introduce Kronecker product operation (denoted by $\otimes$) [16] and according to its property [16], (14) can be transformed into:

$$f(A) = \mathrm{vec}(u^t A v) + b = (v^t \otimes u^t)\mathrm{vec}(A) + b = (v \otimes u)^t \mathrm{vec}(A) + b. \tag{34}$$

where vec(X) denotes an operator that vectorizes a matrix X to corresponding a vector, for example, let $X = (x_{ij}) \in C^{p \times q}$ and $x_i = (x_{1i}, x_{2i}, \ldots, x_{pi})^t$ is its i-th column, thus vec$(X) = (x_1^t, x_2^t, \ldots, x_q^t)^t$ is a $p \times q$ dimensional vector. Comparing (33) with (34), we find that the decision functions in LS-SVM and MatLSSVM have the same form and now, the $v \otimes u$ in MatLSSVM plays the same role as the w in LS-SVM and both have the same dimension. It is easy to prove that the solution space for $v \otimes u$ is contained in that for the w, because usually, the weight vector w of LS-SVM does not *always* satisfy decomposability of the Kronecker product.

Next, we give a set of experiments on two-class datasets to illustrate the relationship between the w and the $v \otimes u$. Due to that the matrixizaion method is difficult to be visually

Table 3 Classification accuracies comparison among MatPCA, MatFLDA, MatLSSVM and MatFLSSVM

Datasets (attributes;classes)	Classifiers			
	MatPCA (%)	MatFLDA (%)	MatLSSVM (%)	MatFLSSVM (%)
Sonar (60A;2)	**80.69 (10 x 6)**	79.54 (3×20)	75.83 (C = 0.25;2×3)	/
Water-T. (38A;2)	82.36 (19×2)	97.20 (19×2)	**98.33 (C = 4;2 x 19)**	/
M.C.2 (166A;2)	80.40 (83×2)	84.14 (83×2)	**87.22 C = 64;83 x 2)**	/
M.C.2 (160A;2)	80.16 (16×10)	79.69 (20×8)	**88.29 C = 0.0625;80 x 2)**	/
ORL (28×23;40)	94.05 (28×23)	94.00 (28×23)	95.30 (C = 2^5;28×23)	**95.35 (C = 32;28 x 23)**
Letter (24×18;10)	87.74 (24×18)	73.60 (24×18)	92.55 (C = 2^{-3};2×216)	**93.10 (C = 0.125;2×216)**
Iris (4A;3)	95.55 (2×2)	95.13 (2×2)	**97.60 (C = 2^0;2 x 2)**	**97.60 (C = 1;2×2)**
Waveform (21A;3)	75.21 (3×7)	76.67 (7×3)	86.05 (C = 2^{-1};3×7)	**86.23 (C = 0.5;3 x 7)**
Wine (12A;3)	81.75 (2×6)	89.14 (4×3)	94.26 (C = 2^2;6×2)	**94.72 (C = 4;6 x 2)**

All the results are gotten at the best regularization constant C (C = 2^{-6}–2^{10}) and corresponding matrix size. The best result for each dataset is denoted in boldface

Table 4 The angle θ between the w and the $v \otimes u$ and the best classification accuracies for each dataset comparison between LS-SVM and MatLSSVM

Dataset (attributes)	Water-T. (38A)	Iris(1,2) (4A)	Sonar (60A)	M.C.2 (160A)	M.C.2 (166A)
θ (in degree)	75.02	18.38	48.53	69.19	70.38
LS-SVM (%)	94.85	100	75.74	88.31	87.78
MatLSSVM (%)	98.33	100	75.83	88.29	87.22

demonstrated in 2D or 3D plot, we can just interpret the obtained results and performance difference by computing the angle between the vector w and the vector $v \otimes u$ (denoted by θ in degree, $\theta \in [-180, 180]$) on the same classification problem. Detail description about the experiment setting is shown in last section. Table 4 shows the best experimental results of LS-SVM and MatLSSVM. From the table, we can observe that for Iris, Sonar, M.C.2 (160A) and M.C.2 (166A) datasets, with the number of the attributes of these datasets growing more, θ also increases larger. However, the differences of the classification accuracies between LS-SVM and MatLSSVM on the above same datasets are comparable and less than 0.56%. From these facts, we believe that in higher dimensional space, although the similarity between the w of LS-SVM and the $v \otimes u$ of MatLSSVM becomes less, LS-SVM and MatLSSVM can still obtain comparable classification effect. In addition, for Water-T. dataset, θ achieves 75.02° and is relatively large, such a difference may be a reason that the best accuracy of MatLSSVM is raised by 3.48% compared with that of LS-SVM. On the whole, a big θ may be a necessary condition of the classification performance difference between LS-SVM and MatLSSVM.

6 Conclusions and Future Work

In this paper, we employed the concept from MatPCA and MatFLDA to develop MatLSSVM. Further, MatFLSSVM, as the fuzzy version of MatLSSVM, was proposed to effectively resolve unclassifiable regions for multi-class problems. And we also reveal the relationship between MatLSSVM and LS-SVM. Differently from the traditional LS-SVM and FLS-SVM operating on vector pattern, they operate directly on matrix patterns and only need to store the two weight vectors with lower dimensionality and the bias such that the so-required storage space is greatly reduced. Through matrixizing the vector patterns, we obtain some useful information for classification and corresponding competitive performance on real world datasets. However, MatLSSVM and MatFLSSVM are both still linear classifiers and their less space demanding only holds for the primal problem (18), i.e., both LS-SVM and FLS-SVM here utilize the linear kernel. Consequently, it is our future work to further extend MatLSSVM and MatFLSSVM to corresponding nonlinear versions and compare them with LS-SVM and FLS-SVM with non-linear kernel.

Acknowledgements This work was partially supported by the Jiangsu Natural Science Foundation Key Project under Grant No. BK2004001 and the Natural Science Foundation of China under Grant No. 60473035, and we also thank the anonymous reviewers very much for their constructive comments.

References

1. Vapnik V (1995) The nature of statistical learning theory. Springer-Verlag, New York
2. Vapnik V (1998) Statistical learning theory. John Wiley, New York

3. Vapnik V (1998) The support vector method of function estimation. In: Suykens, AK, Vandewalle J (eds) Nonlinear modeling: advanced black-box techniques. Kluwer Academic Publishers, Boston, pp 55–85
4. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9:293–300
5. Suykens JAK, Van Gestel T, De Brabanter J, De Moor B, Vandewalle J (2002) Least squares support vector machines. World Scientific, Singapore (ISBN 981-238-151-1)
6. Beymer D, Poggio T (1996) Image representations for visual learning. Science 272:1905–1909
7. Chen SC, Zhu YL, Zhang DQ, Yang JY (2005) Feature extraction approaches based on matrix pattern: MatPCA and MatFLDA. Pattern Recog Lett 26:1157–1167
8. Wang H, Ahuja N (2005) Rank-R approximation of tensors: using image-as-matrix representation. IEEE Conference on Computer Vision and Pattern Recognition, 2005 (CVPR'05)
9. Suykens JAK, De Brabanter J, Lukas L, Vandewalle J (2002) Weighted least squares support vector machines: robustness and sparse approximation. Neurocomputing 48:85–105
10. Yang J, Zhang D, Frangi AF, Yang J-U (2004) Two-dimension PCA: a new approach to appearance-based face representation and recognition. IEEE Trans Pattern Analysis Machine Intelligence 26(1):131–137
11. Ye J, Janardan R, Li Q (2004) Two-dimensional linear discriminant analysis, University of Minnesota. Adv Neural Inform Process Syst 17:1569–1576
12. Li M, Yuan B (2005) 2D-LDA: a statistical linear discriminant analysis for image matrix. Pattern Recognition Lett 26:527–532
13. Tsujinishi D, Abe S (2003) Fuzzy least squares support vector machines for multiclass problems. Neural Netw 16(5–6):785–792
14. KreBel UH-G (1999) Pairwise classification and support vector machines. In: Scholkopf B, Burges CJC, Smola AJ (eds) Advances in kernel methods: support vector learning. MIT Press, Cambridge, MA, pp 255–268
15. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge
16. Graham A (1981) Kronecker products and matrix calculus: with applications. Halsted Press, John Wiley and Sons, NY