Matrix-pattern-oriented least squares support vector classifier with AdaBoost

Zhe Wang Songcan Chen*

Dept of Computer Science & Engineering

Nanjing University of Aeronautics & Astronautics, Nanjing, 210016, China

Abstract: Matrix-pattern-oriented Least Squares Support Vector Classifier (MatLSSVC) can directly classify matrix patterns and has a superior classification performance than its vector version Least Squares Support Vector Classifier (LSSVC) especially for images. However, it can be found that the classification performance of MatLSSVC is matrixization-dependent, i.e. heavily relying on the reshaping ways from the original (vector or matrix) pattern to (another) matrix. Thus, it is difficult to determine which reshaping way is fittest to classification. On the other hand, the changeable and different reshaping ways can naturally give birth to a set of MatLSSVCs with diversity and it is the diversity that provides a means to build an ensemble of classifiers. In this paper, we exactly exploit the diversity of the changeable reshaping ways and borrow AdaBoost to construct an AdaBoost-MatLSSVC ensemble named AdaMatLSSVC. Our contributions are that: 1) the proposed AdaMatLSSVC can greatly avoid the matrixization-dependent problem on single MatLSSVC; 2) different from the ensemble principle of the original AdaBoost that uses a single type of classifiers as its base components, the proposed AdaMatLSSVC is on top of multiple types of MatLSSVCs in different reshapings; 3) since AdaMatLSSVC adopts multiple matrix representations of the same pattern, it can provide a complementarity among different (matrix) representation spaces; 4) AdaMatLSSVC mitigates the selection of the regularization parameter, which are all validated in the experiments here.

Keywords: Vector pattern; Matrix pattern; Least squares support vector classifier (LSSVC); Matrix-pattern-oriented least squares support vector classifier (MatLSSVC); AdaBoost-MatLSSVC ensemble (AdaMatLSSVC); Classifier design; Ensemble system; Pattern recognition.

* Corresponding author: Tel: +86-25-84896481-12106, Fax: +86-25-84498069. Email: s.chen@nuaa.edu.cn (S.C. Chen)

1. Introduction

In statistical pattern recognition learning, the classifier design is one of basic research topics. Generally, the existing classifier designs base on the vector pattern. Therefore, when a pattern under consideration is non-vector, for instance, an image matrix, the matrix first has to be vectorized by concatenating its pixels in some way (Beymer and Poggio, 1996). The vector-pattern-oriented classifier designs indeed bring some convenience for dealing with problems. However, No Free Lunch Theorem (NFL) (Duda et al., 2001) has indicated that it can not be said that one classifier design is always better than another if no prior knowledge can be incorporated, so it is not always effective for the classifier design based on the vector pattern. Especially for images, the vectorization usually leads to a high dimensional image vector space, increases computational complexity (Chen et al., 2000), and even may break down some implicit structural or locally-spatial information among elements of the image (Wang and Ahuja, 2005).

Recently several researchers have paid attention to the problems with the vectorization and independently proposed a technique which can directly operate on matrix pattern without the vectorization preprocessing. Two-dimensional principal component analysis (2DPCA) (Yang et al., 2004) can extract features directly from image matrices, and is shown to be better than classical PCA in favor of both image classification performance and the reduction of computational complexity for feature extraction. Two-dimensional linear discriminant analysis (2DLDA) (Li and Yuan, 2005) also bases on image matrices, overcomes the singularity problem implicitly in classical LDA, and achieves the competitive recognition accuracy on face identification. Further, since both 2DPCA and 2DLDA extract features only from the row direction of image matrices, $(2D)^2$PCA (Zhang and Zhou, 2005), $(2D)^2$FLD (Nagabhushan et al., 2006), Generalized Low Rank Approximations of Matrices (GLRAM) (Ye, 2005) and Non-iterative Generalized Low Rank Approximations of Matrices (NIGLRAM) (Liu and Chen, et al., 2006) were respectively proposed to simultaneously extract features from both the row and column directions of image matrices, and have been proved to improve the performance on both classification and computation. Chen et al. (2005) went further and developed a more general method, called MatPCA and MatFLDA. Compared with the conventional methods such as PCA and LDA, Chen et al.'s method first matrixizes a one-dimensional-vector or image-matrix pattern into its corresponding new matrix pattern before extracting features. In this way, due to that the newly-formed matrix pattern still retains all the original components, the original information generally seems not to

be lost. Further, some new implicit structural or contextual information can likely be additionally introduced in an implicit way.

It has been validated (Chen et al., 2005; Liu and Chen, 2006) that the subsequent classifiers based on the features extracted by the matrixized approaches above, have superior or comparable classification performance to the counterparts designed on the basis of the features obtained by the vector-pattern extractors. But the classifier designs following the matrixized feature extractors still resort to the traditional vector-based technique, that is, the operating pattern of the classifier itself is still of vector representation rather than matrix representation as shown in Figure.1_a. To a certain extent, although the matrix-pattern-oriented feature extraction indeed helps reduce the computational complexity and avoids the possible loss of the information, the vector-pattern-oriented classifier design still needs to vectorize the extracted features that may still have high dimensionality and contain implicit structural or locally-spatial information. The re-vectorization for the extracted features still fails avoiding the high computational complexity and the loss of information possibly. Further, we also expect that the excellent classification performance can be achieved if omitting the feature extraction phase and directly classifying the matrixized pattern. Therefore in doing so, we intentionally evade the feature extraction phase and directly implement the matrix-pattern-oriented classifier design as shown in Figure.1_(b). In practice, we selected least squares support vector classifier (LSSVC) established by Suykens and Vandewalle (1999) as a matrixized paradigm to develop a matrix-pattern-oriented LSSVC version named MatLSSVC (Wang and Chen, 2007). The experiments (Wang and Chen, 2007) verify that the new design is feasible and has the comparable performance to its vector version.

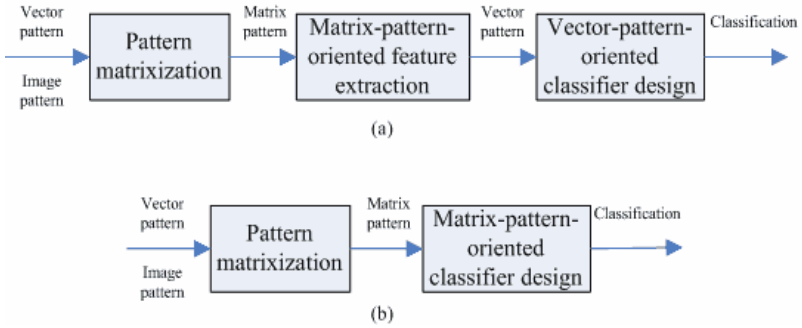

Figure 1.(a) Matrix-pattern-oriented feature extraction + Vector-pattern-oriented classifier;
(b) Matrix-pattern-oriented classifier without feature extraction.

However, it can be found that the classification performance of MatLSSVC is matrixization-dependent, i.e. heavily relying on the different reshaping ways from the original (vector

or matrix) pattern to another matrix pattern. In the process of matrixizing an original one-dimensional or image-matrix pattern, there are different reshaping ways that lead to the matrix pattern with different sizes. Now for convenience, we confine a reshaping mode without overlapping among the components of the pattern, that is, 1) first to partition the vector pattern[1] or image-matrix pattern into the equally-size sub-vectors with one fixed size; 2) then to arrange the generated sub-vectors into the corresponding matrix pattern column-by-column; 3) to repeat 1) and 2) with another size. For example, a vector pattern $A=[1,2,3,4,5,6,7,8]^t$ can be reshaped into two matrix patterns with different sizes: $\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}$ and $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}^t$. In doing so, different matrix representations of the same pattern result in different classification performances. In MatLSSVC (Wang and Chen, 2007), cross validation (CV) is used to choose the fittest matrix pattern representation corresponding to the best classification performance, which causes a large computational complexity. On the other hand, such flexible reshaping ways can exactly give birth to a set of MatLSSVCs with the diversity from the different modes. As a result, in this case, there will be two key problems to be raised: 1) how to deal with the matrixization-dependent problem; 2) how to utilize the diversity simultaneously. In this paper, we try to solve both.

It is well known that the diversity of the component classifiers in an ensemble of classifiers is crucial to boost generalization performance (Dietterich, 2002; Cunningham and Carney, 2000; Krogh and Vedelsby, 1995; Lam, 2000; Littlewood and Miller, 2000; Rosen, 1996). An ensemble of classifiers works by running the base classifier in multiple times, and forming a final decision through combining the component results produced by the base learners. Further, Dietterich (2002) thinks that there are two main approaches for designing an ensemble of classifiers. The first approach is that each component classifier is independently constructed in such a way that their results are accurate and diverse. Typical examples include Bagging (Breiman, 1996), the ensemble of neural networks (Cherkauer, 1996), the combining classifier based on error-correcting output coding (Dietterich and Bakiri, 1995), the random subspace method (Ho, 1998) and so on. The second approach such as AdaBoost (Freund and Schapire, 1996; Freund and Schapire, 1997) is that each component classifier is constructed in a coupled fashion so that the final weighted vote of the classifiers can give a good

[1] If the original pattern is one-dimension (1D), the vector pattern is exactly the original pattern. If the original pattern is 2D, the vector pattern is exactly the one transformed or vectorized from the original 2D pattern in some ways. In this paper, for a 2D pattern, we concatenate its columns one by one.

performance. It is natural to integrate both of the approaches for the better performance. It can be found that MatLSSVCs handling different matrices have both sufficient accuracy and diversity to large extent. By “diversity” here, we mean that for different reshaping ways from an original (vector or matrix) pattern to another matrix, MatLSSVCs have reasonably low but different prediction errors for new samples. Hereby, we try to explore the way of directly integrating different MatLSSVCs into the ensemble designed by the second approach. Since AdaBoost ensemble has been proven excellent and simple (Freund and Schapire, 1996), thus we follow its design principle to construct an AdaBoost-MatLSSVC ensemble (AdaMatLSSVC). However, unlike its original principle that repeatedly runs a *single* base classifier on various distributions over the training set and then combines the component results produced by the base learner into an ensemble, AdaMatLSSVC is based on multiple kinds of MatLSSVCs generated from *different* reshaping ways rather than the *single* kind of MatLSSVC as the base classifiers. According to the combination, we do not have to consider which reshaping way is the fittest for the original pattern. Consequently, to some extent, AdaMatLSSVC mitigates the matrixization-dependent problem on the single MatLSSVC and simultaneously enforces the diversity. Thus it basically overcomes the two key problems raised above.

Further, it can be found that single MatLSSVC can only deal with one matrix representation of the given pattern. One matrix representation corresponds to one representing structural space of the given pattern. Since the proposed AdaMatLSSVC considers multiple different representations of the same pattern, it can provide a complementarity for all the adopted matrix representing structural spaces and lead to a better classification performance than single MatLSSVC, which is validated in our experiments.

The rest of this paper is organized as follows. We review the architecture of MatLSSVC in Section 2. Section 3 gives the proposed AdaMatLSSVC in detail. In Section 4 we present the experimental results and some discussion. Finally, we give the conclusion.

2. Matrix-pattern-oriented classifier design (MatLSSVC)

Given a training set $V = \{x_i, y_i\}_{i=1}^{l}$ with the input patterns $x_i \in R^n$ and the output values $y_i \in \{+1, -1\}$ indicating the class label, the decision function of LSSVC is given by:

$$f(x) = w^t g(x) + b\,, \tag{1}$$

where $g(.)$ is a linear or nonlinear function which maps x into the feature space, w is a weight vector and b is a bias. In this paper, $g(.)$ is only taken as the linear form and thus (1) can be rewritten as:

$$f(x) = w^t x + b, \tag{2}$$

which is a linear decision function directly acting on the vector pattern x.

Then, we attempt to develop a decision function directly operating matrix pattern. Given another training set $M = \{A_i, y_i\}_{i=1}^{l}$ with the input patterns $A_i \in R^{d_1 \times d_2}$ and the output values $y_i \in \{+1, -1\}$ indicating the class labels, the decision function of MatLSSVC is designed as:

$$f(A) = u^t A v + b, \tag{3}$$

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. It is required that for a given pattern A_i, the following condition must be satisfied to the greatest degree:

$$f(A_i) = u^t A_i v + b \begin{cases} \geq 1, & \text{if } y_i = 1; \\ \leq -1, & \text{if } y_i = -1; \end{cases} \quad i = 1, ..., l. \tag{4}$$

Then u, v and b can be got by optimizing the following objective function (Wang and Chen, 2007)

$$\min_{u,v,b,\xi} \quad \frac{1}{2} u^t u + \frac{C}{2} \sum_{i=1}^{l} \xi_i^2, \tag{5}$$

subject to the following equality constraints:

$$y_i (u^t A_i v + b) = 1 - \xi_i, \quad i = 1, ..., l, \tag{6}$$

where C is a regularization constant and ξ_i is the slack variable. The detailed description about MatLSSVC can be found in the literature (Wang and Chen, 2007).

3. The proposed AdaMatLSSVC

There are different reshaping ways that lead to different matrix patterns. For an original one-dimensional or 2D pattern x, MatLSSVC first vectorizes it into $x' \in R^n$, then partitions x' into certain equality-size sub-columns, and column-by-column concatenates the sub-columns into the corresponding matrix $A \in R^{d_1 \times d_2}$ as shown in Figure 2. There will be different As corresponding to different blocks with different sizes in Figure 2 as long as the condition $n == d_1 \times d_2$ is satisfied.

Consequently, MatLSSVCs on different matrix patterns will induce different classification performances. In other words, the changeable reshaping ways can give rise to a set of MatLSSVCs with the diversity, i.e., patterns can be represented by different matrices. In order to mitigate the matrixization-dependence and utilize the diversity, we here use AdaBoost (Freund and Schapire, 1996; Freund and Schapire, 1997) to construct an ensemble (AdaMatLSSVC), which is based on MatLSSVCs in different reshaping ways.

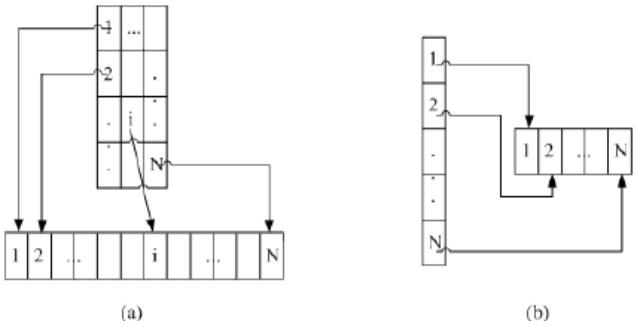

Figure 2: (a) reshaping one image matrix to another; (b) reshaping one original vector to a matrix. (The *i*-th block here denotes a column vector)

Analogously to AdaBoost, our AdaMatLSSVC repeatedly calls the base classifiers in a series of rounds t=1,…,T , where the weight of the distribution on the training sample i on the round t is denoted by $D_t(i)$. However, unlike AdaBoost that only uses a *single* base classifier, AdaMatLSSVC uses *different* MatLSSVCs in different reshaping ways as the base classifiers. The description of AdaMatLSSVC is given in Figure 3. Given $V = \{x_i, y_i\}_{i=1}^{l}$ with $x_i \in R^n$ and $y_i \in Y = \{+1, -1\}$, the initial distribution D_1 is uniform over V so $D_1(i)=1/l$ for all i and the number of the different reshaping ways $S = \{s_j\}_{j=1}^{m}$ is m, where s_j represents the jth reshaping way with its corresponding matrix size written as $d_1^{(j)} \times d_2^{(j)}$ and the value of $d_1^{(j)} \times d_2^{(j)}$ is equal to n. On the round t, we make the *index* equal to *mod*(t, m)+1 and choose the current reshaping way s_j, where *mod*(t, m) denotes the modulus after t divided by m; j is made equal to the *index*; and the *index* ranges in the interval [1, m]. By s_j, we transform $V = \{x_i, y_i\}_{i=1}^{l}$ into $M_t^{(j)} = \{A_i, y_i\}_{i=1}^{l}$, where $x_i \in R^n$ and $A_i \in R^{d_1^{(j)} \times d_2^{(j)}}$. Consequently, AadMatLSSVC provides the base MatLSSVC with the current matrix patterns in $M_t^{(j)}$ and the distribution D_t over the whole training set $M_t^{(j)}(V)$. In response, the base MatLSSVC computes a classifier $h_t : M_t^{(j)} \to R$ which should correctly classify a fraction of the training set that

has the large probability with respect to D_t. That is, given $M_t^{(j)}$, the current base MatLSSVC aims to find an h_t which minimizes the training error $\varepsilon_t = \Pr_{i\sim D_t}[h_t(x_i) \neq y_i]$. This error is measured with respect to the distribution D_t that is provided to the base classifier. This process continues for T rounds. At last, the final classifier h_{fin} is a weight vote of the base classifiers $h_1,\ldots,h_T$. However, a given instance $z \in R^n$ to be classified should be first similarly transformed into the corresponding matrix pattern $z_t \in R^{d_1^{(j)} \times d_2^{(j)}}$ as $M_t^{(j)}$ for each base classifier h_t.

Algorithm 2 –AdaMatLSSVC

Given: $V = \{x_i, y_i\}_{i=1}^{l}$ where $x_i \in R^n$, $y_i \in Y = \{+1, -1\}$.

1. Initialize D_1=1/l; $S = \{s_j\}_{j=1}^{m}$, where s_j represents the jth reshaping way with its corresponding matrix size written as $d_1^{(j)} \times d_2^{(j)}$; the value of $d_1^{(j)} \times d_2^{(j)}$ is equal to n; and m is the number of different reshaping ways.
2. For t=1,…,T:
   a) Let *index*=*mod*(t, m)+1 and choose s_j, where *mod*(t, m) denotes the modulus after t divided by m, j=*index*; and *index* $\in [1, m]$;
   b) By s_j, transform $V = \{x_i, y_i\}_{i=1}^{l}$ into $M_t^{(j)} = \{A_i, y_i\}_{i=1}^{l}$, where $x_i \in R^n$ and $A_i \in R^{d_1^{(j)} \times d_2^{(j)}}$;
   c) Given this $M_t^{(j)}$ dataset, train MatLSSVC using the distribution D_t;
   d) Get the base MatLSSVC $h_t : M_t^{(j)} \to R$;
   e) Calculate the error of h_t: $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$;
   If ε_t>1/2, then set T=t-1 and abort loop;
   f) Set $\beta_t = \varepsilon_t / (1-\varepsilon_t)$;
   g) Update distribution D_t:
   $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & if\ h_t(x_i) == y_i \\ 1 & otherwise \end{cases},$$
   where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution);
3. Output the final classifier:
   $$h_{fin}(z) = \arg\max_{y\in Y}\left(\sum_{t:h_t(z)=y} \log\frac{1}{\beta_t}\right),$$
   where for each base MatLSSVC h_t, the test pattern $z \in R^n$ should be similarly transformed into the corresponding matrix pattern $z_t \in R^{d_1^{(j)} \times d_2^{(j)}}$ as $M_t^{(j)}$.

Figure 3: The algorithm AdaMatLSSVC.

Like AdaBoost, the manner, in which D_t is computed on each round and how h_{fin} is computed in AdaMatLSSVC, may have different ways. We use a similar way like that in (Freund and Schapire, 1996). The details have also been described in Figure 3.

One important theoretical property about AdaBoost is stated in the following theorem. This theorem shows that if the base classifiers consistently have accuracies only slightly better than 1/2, then the training error of the final classifier h_{fin} drops to zero exponentially fast. AdaMatLSSVC only changes the base classifier in the reshaping way and the experiments here indicate that the errors of MatLSSVCs with differently reshaped matrix patterns are less than 1/2 in classification performance. As a result, our AdaMatLSSVC can inherit this original property well.

Theorem 1 (Freund and Schapire, 1997) *Suppose the base learning algorithm* **BaseLearn**, *when called by* **AdaBoost** (**AdaMatLSSVC**), *generates classifiers with errors* $\varepsilon_1,...\varepsilon_t...,\varepsilon_T$, *where* ε_t *is as defined in Figure* 3. *Assume each* $\varepsilon_t \le 1/2$, *and let* $\gamma_t = 1/2 - \varepsilon_t$. *Then the following upper bound holds on the error of the final classifier* h_{fin}*:*

$$\frac{\left|\left\{i : h_{fin}(x_i) \neq y_i\right\}\right|}{l} \le \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \le \exp(-2\sum_{t=1}^{T}\gamma_t^2).$$

4. Experiments

In this section we first explore the diversity of the matrixization on a dataset, then experimentally demonstrate that AdaMatLSSVC can mitigate the matrixization-dependent problem.

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[2], and those in a vector representation: Waveform (21 attributes/3 classes/1500 data)[3], Wine dataset (12/3/178) that is generated through the last one attribute of the original Wine[4] with 13 attributes being omitted (by us) mainly for producing more assembling matrix patterns, Water-treatment (denoted Water-T.) (38/2/116)[4], Sonar (60/2/208)[4], Musk Clean2 (denoted M.C.2) (166/2/6598)[4] and M.C.2 (160/2/6598) that is just the dataset in M.C.2 (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 consisting of the digits 0-9 with each

[2] available at http://sun16.cecs.missouri.edu/pgader/CECS477/NNdigits.zip
[3] available at http://www.ics.uci.edu/~mlearn/MLRository.html
[4] available at ftp://ftp.cs.cmu.edu/afs/cs/project/connect/bench/.

class having 50 samples and each sample is a 24×18 matrix pattern.

Table 1: Benchmark data specification

Data (attributes;classes)	Training set	Test set	T
ORL(28×23;40）	200	200	40
Letter(24×18;10)	250	250	50
Sonar(60;2)	100	108	100
Water-T.(38;2)	50	66	50
M.C.2(166;2)	1000	5598	100
M.C.2(160;2)	1000	5598	100
Waveform(21;3)	750	750	100
Wine(12;3)	48	130	48

4.2. Setting of experiments

In all experiments, each dataset is randomly divided into the two no-overlapping parts with the one for training and the other one for testing. The sizes of both the training and the test sets for all datasets are shown in Table 1. For each classification problem, 10 independent runs are performed on the above partitions and then their classification accuracies on the test sets are averaged and reported. Table 2 shows the abbreviations and explanations of all algorithms that are implemented in our experiments. In addition, the regularization constant C between LSSVC and MatLSSVC, is determined by searching from 2^{-6} through 2^{10} with each step by multiplying 2. For MatLSSVC, v_0 is initialized to $[1_1,\ldots,1_{d2}]^t$; the learning rate η is selected from {1, 10, 100, 1000}; the maximal iterative count *maxIter* is set to 5000 and the ε is set to 10^{-3}. At the same time, the kernel functions corresponding to LSSVC and AdaLSSVC are the linear kernel, i.e. $g(x)=x$. For AdaLSSVC, AdaMatLSSVC and the other ensembles of MatLSSVCs, the regularization constant C is given by the prior knowledge and experience, and the other parameters are given the same as above.

Due to using the AdaBoost technique in AdaLSSVC, E.I.MatLSSVC and AdaMatLSSVC, we choose a set of examples from the original training set at random according to the weight distribution D_t with replacement on each round t, and set the round number T for each dataset as shown in Table 1. Here, it is worth being emphasized that for all the patterns on both ORL and Letter datasets that are 2D matrix themselves, we can still reshape them to another corresponding matrix patterns in different ways. As for the original vector patterns, we can also matrixize them to different matrix patterns in different reshaping ways.

Table 2: The abbreviations and explanations of all algorithms that are implemented in our experiments

Abbreviations	Explanations
LSSVC	least squares support vector classifier
MatLSSVC	the matrixized version of LSSVC
AdaLSSVC	the ensemble that AdaBoost uses LSSVC as the base classifier
M.V.MatLSSVC	the ensemble which combines MatLSSVCs based on different reshaping ways using the majority voting technique
E.I.MatLSSVC	the ensemble that AdaBoost uses individual MatLSSVC in the same reshaping way as the base classifier
AdaMatLSSVC	the ensemble that AdaBoost uses MatLSSVCs in different reshaping ways as the base classifiers

4.3. Results discussion

Table 3 reports the classification accuracies of LSSVC, MatLSSVC, AdaLSSVC, AdaMatLSSVC and M.V.MatLSSVC on different datasets under the above experimental conditions. Both AdaMatLSSVC and M.V.MatLSSVC use those matrix patterns that are written in the third column of Table 3 as combination. In this table, all results are presented in percentages in the columns. For each dataset in Table 3, the best, the second best and the third best results are bolded, underlined and undee-lined, respectively. The regularization constant *C*s of both LSSVC and MatLSSVC are given, by which the best generalization is achieved. For AdaLSSVC, the *C* is equal to that of LSSVC. For AdaMatLSSVC and M.V.MatLSSVC, the *C*s are equal to those of MatLSSVC. Several observations can be made based on this table.

4.3.1. Discussion between MatLSSVC and LSSVC

First of all, MatLSSVC improves the classification performance from slight to distinct, at least for one matrix pattern compared with LSSVC on the four datasets (Letter, Sonar, Water-T., Waveform) and whereas on the other four datasets (ORL, M.C.2(166), M.C.2(160), Wine), the results are opposite. On the whole, compared with LSSVC, MatLSSVC improves performance on 4 out of 8 datasets and especially distinct on Water-T. (achieving about 4%) and on the other hand, on those datasets where MatLSSVC's performance is degraded, the matrixized LSSVC achieves about 2.5% decrease on Wine in the worst case but only slight on the rest. MatLSSVC is competitive with LSSVC in terms of classification performance.

Table 3: Classification accuracy comparison among LSSVC, MatLSSVC, AdaLSSVC, AdaMatLSSVC and M.V.MatLSSVC. The best, the second best and the third best results are bolded, underlined and undee-lined, respectively. The regularization constant *C*s of both LSSVC and MatLSSVC are given where the best generalization is achieved. For AdaLSSVC, the *C* is equal to that of LSSVC. For AdaMatLSSVC and M.V.MatLSSVC, the *C*s are equal to that of MatLSSVC. All these different matrix patterns that MatLSSVC and the ensembles of MatLSSVCs use on each dataset are written in the bracket of the third column.

Datasets (attributes)	Classifiers				
	LSSVC (%)	MatLSSVC (%)	AdaMatLSSVC (%)	M.V. MatLSSVC (%)	AdaLSSVC (%)
ORL (28×23)	96.40(C=2^5)	95.30(C=2^5;28×23) 93.50(C=2^{-5};161×4) 94.00 (C=2^0;92×7) 94.20(C=2^{-1};46×14)	**96.50**	94.10	**96.50(C=2^5)**
Letter (24×18)	92.28 (C=2)	92.55(C=2^{-3};2×216) 91.40(C=2^{-3};4×108) 87.90(C=2^{-3};12×36) 89.40(C=2^{-6};24×18)	**92.96**	91.76	92.60(C=2)
Sonar (60)	75.74(C=2)	75.83(C=2^{-2};2×30) 74.35(C=2^{-1};3×20) 75.46(C=2^{-1};6×10)	**79.26**	73.70	78.70(C=2)
Water-T. (38)	94.85(C=2^{-6})	96.21(C=2^8;19×2) 98.33(C=2^2;2×19)	**98.79**	93.48	97.42(C=2^{-6})
M.C.2 (166)	87.78(C=2^4)	87.22(C=2^6;83×2) 86.85(C=2^{-5};2×83)	87.98	86.99	**88.33(C=2^4)**
M.C.2 (160)	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)	88.71	81.81	**88.81(C=2^{-2})**
Waveform (21)	86.01(C=2^4)	86.05(C=2^{-1};3×7) 85.20(C=2;7×3)	**86.53**	85.33	86.37(C=2^4)
Wine (12)	96.98(C=2^{-3})	94.26(C=2^2;6×2) 90.00(C=2^{-1};2×6) 89.25(C=2^{-2};4×3) 90.00(C=2^{-2};3×4)	96.23	91.04	**97.17**(C=2^{-3})

Further, from the experimental results, it can be also found that for the same dataset, the differently-reshaped matrices result in the different classification performance. For example, by reshaping the original size 24×18 matrix of Letter to the new 2×216 matrix, MatLSSVC gets the better classification performance on the 2×216 matrix. Before exploration, we first reformulate the discrimination function (3) of the matrix-pattern-oriented classifier (MatLSSVC) and let $B=A^t u$, then (3) can be rewritten as $f(B)=B^t v+b=v^t B+b$. Formally, $f(B)$ is a similar form as (2) that of the vector-pattern-oriented classifier (LSSVC) in which the B is an input to the vector-pattern-oriented

classifier (LSSVC). We decompose B in terms of

$$B = A^t u = [a_1, ..., a_{d_2}]^t u = [a_1{}^t u, ..., a_{d_2}{}^t u]^t, \quad (7)$$

where $A=[a_1,...,a_i,...,a_{d_2}], a_i=[a_{1i},...,a_{d_1 i}]^t, i=1,2,...d_2; B=[b_1,...,b_j,...,b_{d_2}]^t, b_j = a_j^t u, j=1,2,...,d_2$. Thus, all the components of each column of the original matrix A are linearly combined to each component of the new input B, implying that it integrates global information in each column (coined column-global information) and thus de-emphasizes local information in each column (coined column-local information). Instead, if we make r smaller than d_1, reshape the original pattern A with dimension $d_1 \times d_2$ to a new matrix pattern C with dimension $r \times c$, and then still let $B=C^t u$, similarly to the above analysis, all the components of each column of C linearly are combined to each component of the B. Now due to that r is smaller than d_1, in this case, each column of the C is a sub-column of the original pattern A, and thus all the components of a sub-column of A are linearly combined to each component of the B, implying that it just integrates column-local information rather than column-global information and thus emphasizes local information in each column. Consequently, we can obtain the fact that the reshaping from one matrix pattern to another may destroy the whole or global structure in each column of the original pattern, but partial or local structure or column-local information can likely be kept and emphasized contrarily, which could be more useful for discrimination (Maree et al., 2005). Such a fact is the possible reason of yielding the above phenomenon in image recognition. In a word, compared with the vector-pattern-oriented one (LSSVC), the matrix-pattern-oriented classifier (MatLSSVC) may get the guide by a priori knowledge of the specific problem but in an implicit way. And the matrix-pattern-oriented classifier (MatLSSVC) becomes more flexible and effective for classification due to the incorporation of both the matrixization and the reshaping to vector and image-matrix patterns.

4.3.2. Comparison among MatLSSVC, AdaLSSVC, AdaMatLSSVC and M.V.MatLSSVC

From the different performance exhibitions for different matrix patterns on the same dataset, the performance of MatLSSVC is matrixization-dependent, i.e., heavily relying on the reshaping ways from an original (vector or matrix) pattern to another matrix one. On the other hand, due to utilizing the diversity of the matrixization, it can be found that compared with MatLSSVC, our AdaMatLSSVC outperforms all MatLSSVCs with these different matrix patterns on all datasets, which exactly validates the complementarity induced by multiple matrix representations of the same pattern. One

matrix representation only describes one structural space for a given pattern. Since the proposed AdaMatLSSVC considers multiple structural spaces of the same pattern, it can provide a complementarity for each space and have a superior classification performance to MatLSSVC that only considers one structural space.

Finally, in order to further investigate the performance of AdaMatLSSVC, it is also compared with both AdaLSSVC and M.V.MatLSSVC. According to Table 3, the classification accuracies of AdaMatLSSVC are distinctly better than those of M.V.MatLSSVC on all the datasets. Compared with AdaLSSVC, the accuracies of AdaMatLSSVC are better on the four datasets (Letter, Sonar, Water-T., Waveform), comparable on the ORL dataset, worse on the three remaining datasets (M.C.2(166), M.C.2(160), Wine). AdaMatLSSVC is also competitive with AdaLSSVC like the relationship between LSSVC and MatLSSVC.

4.4. Further exploration of AdaMatLSSVC

As the above section describes, AdaMatLSSVC can mitigate the matrixization-dependent problem with MatLSSVC. The reason behind possibly attributes to that AdaMatLSSVC bears the other two important factors besides MatLSSVC itself: the one is the AdaBoost technique and the other is the diversity induced by the different reshaping ways. We have found that AdaMatLSSVC distinctly outperforms M.V.MatLSSVC with the same diversity, which demonstrates that AdaBoost technique plays an important role in improving the performance of AdaMatLSSVC. Further, we need to explore the role of the diversity in improving the performance of AdaMatLSSVC and so implement E.I.MatLSSVC that uses MatLSSVC only in a single reshaping way as the base classifier. Here, there are two kinds of E.I.MatLSSVCs in the corresponding reshaping ways for each dataset: the one uses the reshaping way that has the best classification accuracy in MatLSSVC on each dataset and the other uses the one that has the worst in MatLSSVC.

Table 4 reports these classification accuracies of MatLSSVC, AdaMatLSSVC and E.I.MatLSSVC on all datasets. Like Table 3, AdaMatLSSVC uses the matrix patterns that are written in the second column of Table 4 as combination. For each dataset, the best, second best and third best results are still bolded, underlined and undee-lined, respectively. First, we set the *C*s in both AdaMatLSSVC (thethird column) and E.I.MatLSSVC to equal that of MatLSSVC. Clearly, AdaMatLSSVC distinctly outperforms E.I.MatLSSVC on all the datasets: both ORL and M.C.2 (166) increasing in performance by more than 2%, the other datasets increasing in performance by nearly 1~2%. Further, compared

Table 4: Classification accuracy comparison among MatLSSVC, AdaMatLSSVC and E.I.MatLSSVC. The best, the second best and the third best results are bolded, underlined and undee-lined, respectively. The regularization constant *C* of MatLSSVC is given where the best generalization is obtained. There are two AdaMatLSSVCs that use different *C*s. The one uses the *C* that is equal to that of MatLSSVC. The other one uses the *C* that is set to the prior value 2^{-3}. For E.I.MatLSSVC, the *C* is equal to that of MatLSSVC. All these different matrix patterns that MatLSSVC and AdaMatLSSVC use on each dataset are written in the bracket of the second column.

Datasets (attributes)	Classifiers			
	MatLSSVC (%)	Ada-MatLSSVC (%)	E.I.MatLSSVC (%)	Ada-MatLSSVC ($C=2^{-3}$;%)
ORL (28×23)	95.30($C=2^5$;28×23) 93.50($C=2^{-5}$;161×4) 94.00 ($C=2^0$;92×7) 94.20($C=2^{-1}$;46×14)	**96.50**	93.60($C=2^5$;28×23) 93.05($C=2^{-5}$;161×4)	96.00
Letter (24×18)	92.55($C=2^{-3}$;2×216) 89.40($C=2^{-6}$;24×18) 91.40($C=2^{-3}$;4×108) 87.90($C=2^{-3}$;12×36)	**92.96**	92.04($C=2^{-3}$;2×216) 91.20($C=2^{-6}$;24×18)	92.84
Sonar (60)	75.83($C=2^{-2}$;2×30) 74.35($C=2^{-1}$;3×20) 75.46($C=2^{-1}$;6×10)	**79.26**	77.78($C=2^{-2}$;2×30) 75.93($C=2^{-1}$;3×20)	77.04
Water-T. (38)	96.21($C=2^8$;19×2) 98.33($C=2^2$;2×19)	**98.79**	97.42($C=2^8$;19×2) 98.64($C=2^2$;2×19)	98.64
M.C.2 (166)	87.22($C=2^6$;83×2) 86.85($C=2^{-5}$;2×83)	**87.98**	85.22($C=2^6$;83×2) 85.49($C=2^{-5}$;2×83)	87.56
M.C.2 (160)	88.29($C=2^{-4}$;80×2) 82.31($C=2^0$;10×16) 87.00($C=2^{-5}$;2×80) 84.47($C=2^{-4}$;16×10)	**88.71**	87.17 ($C=2^{-4}$;80×2) 87.62($C=2^0$;10×16)	88.30
Waveform (21)	86.05($C=2^{-1}$;3×7) 85.20(C=2;7×3)	**86.53**	85.73($C=2^{-1}$;3×7) 85.12(C=2;7×3)	86.27
Wine (12)	94.26($C=2^2$;6×2) 89.25($C=2^{-2}$;4×3) 90.00($C=2^{-1}$;2×6) 90.00($C=2^{-2}$;3×4)	**96.23**	94.81($C=2^2$;6×2) 94.43($C=2^{-2}$;4×3)	95.28

with MatLSSVC, E.I.MatLSSVC wins only on the three datasets (Sonar, Water-T., Wine) but loses on the five datasets (ORL, Letter, M.C.2(166), M.C.2(160), Waveform). Consequently, the diversity in AdaMatLSSVC also plays a greatly important role in improving the performance. Because setting the same *C* in both AdaMatLSSVC and MatLSSVC results in more time cost, thus instead, we directly set the *C* of AdaMatLSSVC to the prior value 2^{-3}. Although AdaMatLSSVC with (arbitrarily set) $C=2^{-3}$ has worse performance than that with the *C* value optimally searched in a given range, it still

outperforms single MatLSSVC with optimal parameters. Thus the proposed AdaMatLSSVC seems not relatively sensitive to the regularization parameter *C* than MatLSSVC.

5. Conclusions

We have shown that MatLSSVC, as a matrixized LSSVC version, has the comparable classification performance to LSSVC. However, MatLSSVC's classification performance depends on the different reshaping ways for each pattern of each dataset. The different reshaping ways lead to the diversity of the designed classifiers. Thus through using the AdaBoost technique and the diversity, we develop AdaMatLSSVC that is an ensemble constructed by MatLSSVCs in different reshaping ways as the base classifiers. The experimental results demonstrate that 1) AdaMatLSSVC can largely bypass the matrixization-dependent problem in MatLSSVC; 2) AdaMatLSSVC can provide a complementarity among different reshapings and have a relatively superior classification performance to MatLSSVC; 3) AdaMatLSSVC relaxes the selection of the regularization parameter. Furthermore, according to these results, it can also be observed that both the AdaBoost technique and the diversity of the reshaping ways play their corresponding important roles in improving classification performance.

References

Beymer D., Poggio T., 1996. Image representations for visual learning. Science 272, 1905-1909.

Breiman L., 1996. Bagging predictor. Machine Learning 24, 123-140.

Chen L.F., Liao H.Y.M., Ko M.T., Lin J.C., Yu, G.J., 2000. A new LDA-based face recognition system which can solve the small sample size problem. Pattern Recognition 33, 1713-1726.

Chen S.C., Zhu Y.L., Zhang D.Q., Yang J., 2005. Feature extraction approaches based on matrix pattern: MatPCA and MatFLDA. Pattern Recognition Letters 26, 1157-1167.

Cherkauer K.J., 1996. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In Working Notes of the AAAI Workshop on Integrating Multiple Learned Modes, (P. Chan, Ed.), Menlo Park: AAAI Press, 15-21.

Cristianini N., Shawe-Taylor J., 2000. An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, Cambridge.

Cunningham P., Carney J., 2000. Diversity versus quality in classification ensembles based on feature selection. Technical Report TCD-CS-2000-02, Department of Computer Science, Trinity College Dublin.

Dietterich T.G., 2002. Ensemble learning. In The Handbook of Brain Theory and Neural Networks, second edition.

Dietterich T.G.., Bakiri G., 1995. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2, 263-286.

Duda R.O., Hart P.E., Stock D.G., 2001. Pattern Classification, 2nd Edition. New York: John Wiley and Sons, Inc.

Freund Y., Schapire R.E., 1996. Experiments with a new boosting algorithm. In Machine Learning: Proceedings of the Thirteenth International Conference, 148-156.

Freund Y., Schapire R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55(1), 119-139.

Graham A., 1981. Kronecker Products and Matrix Calculus: with Applications. Halsted Press, John Wiley and Sons, NY.

Ho T.K., 1998. The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 832-844.

Krogh A., Vedelsby J., 1995. Neural network ensembles, cross validation and active learning. In G. Tesauro, D. Touretzky, T. Leens (Eds.), Advances in neural information processing systems, Cambridge, MA: MIT Press, 7, 231-238.

Lam L., 2000. Classifier combinations: Implementations and theoretical issues. In J. Kittler and F. Roli (Eds.), Multiple classifier systems, Vol. 1857 of Lecture Notes in Computer Science, Cagliari, Italy, Springer, 78-86.

Li M., Yuan B., 2005. 2D-LDA: A statistical linear discriminant analysis for image matrix. Pattern Recognition Letters 26, 527-532.

Littlewood B., Miller D., 2000. Conceptual modeling of coincident failures in multiversion software. IEEE Transactions on Software Engineering 15(12), 1596-1614.

Liu J., Chen S.C., 2006. Non-iterative generalized low rank approximation of matrices. Pattern Recognition Letters 27(9), 1002-1008

Maree R., Geurts P., Piater J., Wehenkel L., 2005. Random subwindows for robust image classification. In: Proc. Of IEEE Conf. on Computer Vision and Pattern Recognition.

Nagabhushan P., Guru D.S., Shekar B.H., 2006. $(2D)^2$FLD: An efficient approach for appearance based object recognition. Neurocomputing 69, 934-940.

Rosen B., 1996. Ensemble learning using decorrelated neural networks. Connection Science 8(3/4), 373-383.

Suykens J.A.K., Vandewalle J., 1999. Least Squares Support Vector Machine Classifiers. Neural Processing Letters 9, 293-300.

Vapnik V., 1995. The Nature of Statistical Learning Theory. Springer-Verlag, New York.

Vapnik V., 1998. Statistical learning theory. John Wiley, New York.

Vapnik V., 1998. The support vector method of function estimation. In J.A.K. Suykens and J. Vandewalle (Eds) Nonlinear Modeling: Advanced Black-Box Techniques, Kluwer Academic Publishers, Boston, 55-85.

Wang H., Ahuja N., 2005. Rank-R Approximation of Tensors: Using Image-as-Matrix Representation. IEEE Conf. on Computer Vision and Pattern Recognition.

Wang Z., Chen S.C., 2007. New Least Squares Support Vector Machines Based on Matrix Patterns. Neural Processing Letters 26:41–56.

Yang J., Zhang D., Frangi A.F., Yang J.U., 2004. Two-Dimension PCA: A New Approach to Appearance-Based Face Representation and Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(1), 131-137.

Ye J., 2005. Generalized low rank approximation of matrices. Machine Learning 61(1-3): 167-191.

Zhang D.Q., Zhou Z.H., 2005. $(2D)^2$PCA: Two-directional two-dimensional PCA for efficient face representation and recognition. Neurocomputing 69, 224-231.