Feature Extraction Approaches Based On Matrix Pattern: MatPCA and MatFLDA

Songcan Chen [1*] Yulian Zhu [1] Daoqiang Zhang [1] Jing-Yu Yang [2]

[1]Dept. of Computer Science and Engineering, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, People's Republic of China.

[2]Dept. of Computer Science, Nanjing University of Science & Technology, Nanjing 210094, People's Republic of China

Abstract

Principle component analysis (PCA) and Fisher linear discriminant analysis (FLDA), as two popular feature extraction approaches in Pattern recognition and data analysis, extract so-needed features directly based on vector patterns, i.e., before applying them, any non-vector pattern such as an image is first vectorized into a vector pattern by some technique like concatenation. However, such a vectorization has been proved not to be beneficial for image recognition due to consequences of both the algebraic feature extraction approach and 2DPCA. In this paper, inspired by the above two approaches, we try an opposite direction to extract features for any vector pattern by first matrixizing it into a matrix pattern and then applying the matrixized versions of PCA and FLDA, MatPCA and MatFLDA, to the pattern. MatFLDA uses, in essence, the same principle as the algebraic feature extraction approach and is constructed in terms of similar objective function to FLDA while MatPCA uses a minimization of the reconstructed error for the training samples like PCA to obtain a set of projection vectors, which is somewhat different derivation from 2DPCA despite of equivalence. Finally experiments on 10 publicly obtainable datasets show that both MatPCA and MatFLDA gain performance improvement in different degrees respectively on 7 and 5

[*] Corresponding author: Tel: +86-25-489-2805; +86-25-489-3777. E-mail: s.chen@nuaa.edu.cn (S.C. Chen). lianyi_1999@sohu.com (Y.L. Zhu), daoqz@nuaa.edu.cn.com (D.Q. Zhang), yangjy@mail.njust.edu.cn (J-Y. Yang)

datasets and at the same time, the computational burden of extracting features is largely reduced. In addition, it is noteworthy that the proposed approaches are still linear and the promotion of classification accuracy does not result from commonly-used non-linearization for the original linear approaches but from the simple matrixization. Furthermore, another prominent merit of matrixizing FLDA is that it can naturally break down the notorious rank limitation, that is, the number of discriminating vectors able to be found is bounded by C-1 for C class problem, and at the same time no additional computational cost is introduced.

Keywords: Pattern representation; Principal component analysis (PCA); Fisher linear discriminant analysis (FLDA); Vector representation; Matrix representation; Feature extraction; Pattern recognition.

1. Introduction

PCA (Mclachlan, 1992) and FLDA (Fukunaga, 1990) are two popular approaches to feature extraction and have widely been applied in Pattern recognition and dada analysis. Both usually manipulate directly on a vector pattern, therefore, when a pattern itself to be processed is an image, for instance, a face, the image first has to be transformed or vectorized into a vector pattern by concatenating its pixels in some way and then PCA or FLDA can be applied onto it. This type of vector representation is natural in most of data analyses (Beymer and Poggio, 1996). However, in fact, there also are indeed several extracting feature methods directly from original image matrix rather than its vectorized counterpart such as 2D fast Fourier or discrete cosine transformations (FFT or DCT) (Duhamael and Vetterli, 1990, Clarke, 1985), 2D wavelet transformation (Mallat, 1989) and so on. Nevertheless, they use the different principles from both PCA and FLDA based on the matrix algebra theory. Intuitively, directly manipulating images and extracting their features by means of the matrix algebra method seems simpler and also not to lose too much spatial or structural information of the original images. Several researchers have made such attempts along this line. Hong (1991) employed singular value decomposition (SVD) to extract a set of singular values as classification features directly from the image matrix and a good performance of classification is obtained on their own face base. Tian *et al* (2003) further developed and refined Hong's work and finally obtained better classification performance on some public face datasets. Instead of extracting SV features from single given image, Liu *et al* (1993) directly used a set of given training image matrices to construct an optimal discriminant criterion similar to FLDA and subsequently obtained so-needed discriminant features for classification. As a result, a good classification performance was achieved on their own collecting face base and thus an image FLDA (IMFLDA, although the authors did not name so) was developed. And

at the same time, Yang *et al* extended the other classical algebraic method of extracting feature, PCA, to the image PCA (IMPCA) (Yang and Yang, 2002) or 2-dimensional PCA (2DPCA) (Yang, Zhang, Frangi and Yang, 2004). Unlike PCA of minimizing the sample reconstructed error, IMPCA or 2DPCA is maximizing a sample scatter measure. Through the experiments on several well-known benchmark face bases, 2DPCA or IMPCA was shown to be better in favor of both image classification and reduction of computation complexity for feature extraction. Inspired by their successes, in this paper, we will make an opposite attempt of whether IMFLDA and IMPCA can also be applied to a vector pattern. Our motive is from such an intuitive observation: if a vector pattern is matrixized or assembled into a corresponding matrix pattern, then we apply the matrixizied PCA and FLDA to extract their features, which possibly also facilitates both raise of classification performance and reduction of computational time because , 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. Therefore, the objectives of this paper are twofold: 1) tailor both PCA and FLDA to directly handle the matrix pattern, that is, to extract features from the matrix pattern and at the same time still to retain their simplicity and effectiveness by means of the principle of the 2DPCA (or IMPCA) (Yang and Yang, 2002)(Yang, Zhang, Frangi and Yang, 2004) and generalized IMPCA (GIMPCA) (Chen and Liang, 2002); 2) investigate the usefulness of matrixizing a vector pattern and compare their classification accuracies respectively on several publicly obtained real world and artificial datasets.

The rest of this paper is organized as follows: In Section 2, we will detail PCA and FLDA operating directly on matrixized patterns and call them MatPCA and MatFLDA, respectively. Their formulations illustrate that the traditional PCA and FLDA are just corresponding special cases of the

new approaches. In Section 3, we present our experimental results on several public benchmark datasets and point out that in their recognition accuracies, the method using the matrix representation on some datasets is better and more efficient than that with the vector representation. We conclude in Section 4.

2. MatPCA and MatFLDA

For feature extraction and subsequent recognition, in this section, we give two methods, MatPCA and MatFLDA, of extracting features directly from matrix patterns including images, re-assembled images (Chen and Liang, 2002) and matrixized patterns from other vector patterns.

2.1 MatPCA

2.1.1 Description of MatPCA

An idea of constructing MatPCA is from image PCA (IMPCA) or 2DPCA, developed especially for extracting features directly from image. Hereafter, for both unification of these names and similarity of description, we will rename IMPCA or 2DPCA and GIMPCA (Chen and Liang, 2002) as MatPCA and describe below:

Suppose we are given M m-by-n matrix pattern A_j (j=1,2,…, M) available, and their mean is denoted by $\overline{A}$. Let x be a projection vector with m components. MatPCA tries to project a matrix pattern A onto a set of projection vectors $X = [x_1, x_2, \cdots, x_d]$ satisfying the constrains $X^T X = I$ (I an identity matrix and T denotes transpose of matrix) by the following *linear* transformation

$$Y = X^T (A - \overline{A}) . \qquad (1)$$

where $Y = [y_1^T, y_2^T, \cdots, y_d^T]^T$, an extracted d-by-n feature matrix with each row vector $y_i (i = 1,2,\cdots,d)$ meeting the equations: $y_i = x_i^T (A - \overline{A})$, and d is the number of the projection directions to be found. Thus for each $A_i, i = 1,2,\cdots,M$, we have

$$Y_i = X^T (A_i - \overline{A}) . \quad (2)$$

Now we wish to construct such a criterion that it can retain as much original information of the training set in the projected space as possible by optimizing the criterion. One of the criteria satisfying such a requirement is the reconstructed error (RCE) of all the training samples like in PCA. More specifically, minimizing the following criterion:

$$RCE(X) = \frac{1}{M} \sum_{i=1}^{M} \left\| A_i - \hat{A}_i \right\|^2 \quad (3)$$

we can get X. Here $\|\cdot\|$ denotes the matrix 2-norm and equivalently can be rewritten as $\|A\|^2 = tr(AA^T)$ ($tr()$ is a matrix trace operation), and $\hat{A}_i = XY_i + \overline{A}$ is a reconstruction for A_i (i=1, 2, …, M). Substituting all the reconstructions and (2) into (3) and after some manipulation, we have

$$RCE(X) = \frac{1}{M} \sum_{i=1}^{M} \left\| A_i - \hat{A}_i \right\|^2 = \frac{1}{M} \sum_{i=1}^{M} \left\| A_i - \overline{A}) \right\|^2 - tr(X^T S_t^{Mat} X)$$

where $S_t^{Mat} \overset{def}{=} \frac{1}{M} \sum_{i=1}^{M} \left(A_i - \overline{A}\right)\left(A_i - \overline{A}\right)^T$, called *matrix total covariance Matrix* or *total covariance matrix constructed by given sample matrices*, and formally completely similar to the scatter matrix in PCA and thus why we call it as matrix PCA (MatPCA). And it is easy to verify that it is positive semi-definite and has all non-negative eigenvalues.

Obviously the first term of $REC(X)$ is a constant for given samples, so the minimization of $REC(X)$ equivalently maximizes $J(X)$ below

$$J(X) = tr(X^T S_t^{Mat} X) \quad (4)$$

This is an original definition of Yang *et al* (2002, 2004) deriving 2DPCA. Furthermore, maximizing (4) under the constraints $X^T X = I$ equivalently solves the following eigenvalue-eigenvector matrix equation

$$S_t^{Mat} X = X\Lambda \quad (5)$$

Here $\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_m)$ is a diagonal matrix consisting of all non-negative eigenvalues

associating with the eigenvector matrix X. In order to keep as much original information as possible while compressing the original information, we just take the first d eigenvectors (which compose X) with respect to the first d largest eigenvalues, where d is determined in terms of $\frac{\sum_{i=1}^{d}\lambda_i}{\sum_{i=1}^{m}\lambda_i} \geq \theta$ (in general, taken as 98%), and then use them to project any matrix pattern to produce a new feature matrix Y and finally apply the Y to classify the unknown matrix pattern based on the nearest neighbor rule (NN) (Loizou and Maybank, 1987).

Since the scale of the *matrix total covariance matrix* in MatPCA is generally much smaller than that of PCA based on the vector representation, much computational time will be saved as shown in (Yang and Yang, 2002), (Yang, Zhang, Frangi and Yang, 2004), (Chen and Liang, 2002). Taking the Musk-Clean2 data with 160 dimensions an example, in PCA, the scale of the scatter matrix is 160x160, and in MatPCA, when the data is matrixized to a 10x16 or 16x10 matrix pattern (refer to Sub-section 2.3), the scale of the *matrix total covariance matrix* is just 10x10 or 16x16 respectively. The compression ratios of both reach 1/256 and 1/100, respectively. Such a reduction is naturally very attractive.

2.1.2 Comparisons with PCA

Except for time-savings and still being a linear approach, MatPCA has also other features different from PCA:

1) It is a generalized PCA

From (1), we can easily derive the PCA when the A itself is a vector pattern or a vectorized pattern from matrix pattern, in other words, PCA is just a special case of the MatPCA.

2) It is a PCA with multiple clusters

By decomposing (1), we can rewrite it in the component-wise form:

$$y_j = X^T a_j, j = 1,2,\cdots,n \tag{6}$$

where $A = [a_1, a_2, \cdots, a_n]$, a_j is the jth column of A and n is its column number. If each column is viewed as a vector sub-patterns, then the whole training samples will consist of nM m-dimensional vector patterns. Accordingly, the sample mean can be decomposed as $\overline{A} = [\overline{a}_1, \overline{a}_2, \cdots, \overline{a}_n]$ with each $\overline{a}_j, j = 1,2,\cdots,n$ being the jth column of $\overline{A}$. From these, the *matrix total scatter matrix* can be formulated as (7)

$$\begin{aligned} S_t^{Mat} &= \frac{1}{M}\sum_{i=1}^{M}\left(A_i - \overline{A}\right)\left(A_i - \overline{A}\right)^T \\ &= \sum_{j=1}^{n}\left[\tfrac{1}{M}\sum_{i=1}^{M}\left(a_{ij} - \overline{a}_j\right)\left(a_{ij} - \overline{a}_j\right)^T\right] \\ &= \sum_{j=1}^{n} S_{tj}^{Col} \end{aligned} \tag{7}$$

where $S_{tj}^{Col}, j = 1,2,\cdots,n$ define the n column scatter matrices with m-by-m dimension and are positive semi-definite. So the present scatter matrix denoted by S_t^{Matl} actually is based on the n cluster centers rather than single total sample center or mean as in PCA. Possibly, it is this very point that mines additional information out of the matrix pattern to make final classification performance improvement.

2.2 MatFLDA

Using similar technique to deriving MatPCA, in this subsection, we describe FLDA. As an unsupervised method, PCA does not use any class information, which aims simply at preserving as much information of the original whole samples as possible. In contrast, FLDA is a supervised method and uses implicitly given class information to extract pattern features. Now suppose there are C matrix pattern classes available, $\varpi_i = \{A_{ij}\}_{j=1}^{N_i}, i = 1,2,\cdots,C$ denotes the ith class, their class means are, respectively, $\overline{A_i} = \frac{1}{N_i}\sum_{j=1}^{N_i} A_{ij}, i = 1,2,\cdots,C$, and the total sample mean is still defined as $\overline{A}$. Let x

be a vector with m components. MatFLDA attempts to project a matrix pattern A onto the x satisfying the constraint that $x^T x = 1$ by the following *linear* transformation

$$y = x^T A. \tag{8}$$

where y is an extracted feature matrix or projected value with the size of $1 \times n$. Thus for each $A_{ij}, i = 1,2,\cdots,C; j = 1,2,\cdots,N_i$, all their projected values can be found in terms of

$$y_{ij} = x^T A_{ij}, i = 1,2,\cdots,C; j = 1,2,\cdots,N_i. \tag{9}$$

Similar to FLDA, in order to obtain an optimal discriminating vector x, we define the following objective function and maximize it:

$$J_{Mat}(x) = \frac{tr(x^T S_b^{Mat} x)}{tr(x^T S_w^{Mat} x)} \tag{10}$$

where

$$S_b^{Mat} = \sum_{i=1}^{C} N_i \left(\overline{A_i} - \overline{A}\right)\left(\overline{A_i} - \overline{A}\right)^T \tag{11}$$

is the *total between-class scatter matrix* and

$$S_w^{Mat} = \sum_{i=1}^{C} \sum_{j=1}^{N_i} \left(A_{ij} - \overline{A_i}\right)\left(A_{ij} - \overline{A_i}\right)^T \tag{12}$$

the *total within-within scatter matrix*. Both are formally identical to the corresponding scatter matrices in FLDA and it is easy to verify that they are both positive semi-definite or positive definite. By maximizing $J_{Mat}(x)$, we expect to achieve two points in the projected space: one is to make the within-class spread or scatter as small as possible; and the other is the between-class spread or scatter as large as possible. In this way, the projection vector x can maximally embody so-needed discriminating information. Differentiating $J_{Mat}(x)$ with respect to x under the constraint $x^T x = 1$ and zeroing the derivative, we can derive the following generalized eigenvalue-eigenvector equation that x satisfies:

$$S_b^{Mat} x = \lambda S_w^{Mat} x \tag{13}$$

In order to get more discriminating information from the original samples, generally we need to seek as many projection vectors as possible as done in MatPCA. As we have known, the traditional FLDA can only obtains at most *C*-1 projections for the *C* class problem because S_b^{Mat} is the sum of the *C* outer-products of the vector patterns (at this time, the matrix pattern has been vectorized to the corresponding vector pattern) and thus its rank is at most *C*-1 no matter how large the vector dimensionality due to the condition that $\sum_{i=1}^{C} N_i \left(\overline{A_i} - \overline{A}\right) = 0$. Such a rank limitation is unfavorable and results in some difficulties of mining more discriminating information from data. In order to relax the limitation, some variants of FLDA have been proposed to avoid part of the problem. For example, besides the commonly used maximum likelihood (ML) estimator of S_w^{Mat} in the vector representation, various regularization techniques are available to obtain its robust estimates (Friedman, 1989) (Hastie, Buja and Tibshirani, 1995). In (Okada and Tomita, 1985), the limitation was further lifted by selecting the projection vector one at a time under an orthogonality constraint. Surprisingly, S_b^{Mat} in the matrix representation seems to naturally have such an ability to relax the limitation because its rank approaches maximally to *m*, where *m* is the rank of $\left(\overline{A_i} - \overline{A}\right)\left(\overline{A_i} - \overline{A}\right)^T, i = 1,2,\cdots,C$ and can be taken a value greater than *C*-1 by assembling. In this way, the number of the projections found by MatFLDA can usually be ensured to be greater than that found by FLDA. In sum, for MatFLDA, we have 1) it is a generalized FLDA; 2) it is a FLDA with multiple clusters besides using the original class means; 3) it can breakdown the limitation of rank.

Now we can take the first *d* eigenvectors (which compose *X*) with best discriminating ability with respect to the first *d* largest generalized eigenvalues as described by (13) and use them to project any unknown matrix pattern to produce a new feature matrix *Y* (as in MatPCA) and finally use the *Y* to

classify the unknown matrix pattern by means of the nearest neighbor rule (NN). Hereafter, for convenience of notation, we will not anymore differentiate MatPCA and MatFLDA with MatPCA and MatFLDA incorporating NN classifier.

2.3 Matrixization of vector pattern

Matrixization of a vector pattern is the first and also a key step of performing MatPCA and MatFLDA and will lead to lots of reductin on the computational times. The following is a brief description.

1) If a pattern itself is a matrix pattern such as a face image, then generally we do not have to matrixize or re-assemble it to another matrix pattern and instead directly present it to both MatPCA and MatFLDA despite such matrixization or re-assembling is also allowed, for example, block partition in image compression.

2) If an original pattern is a vector, then whether the matrixization needs to perform or not depends on using FLDA and PCA or MatFLDA and MatFLDA. We will here use an assembling mode without overlapping among pattern components, that is, the pattern (a column vector) is partitioned into several equally-sized sub-patterns or column sub-vectors, and then arranged column-by-column into a corresponding matrix called a matrix pattern. For example, a vector pattern $A = [1,2,3,3,5,2,7,0,2,1]^T$ can be assembled into $\begin{bmatrix} 1 & 3 & 5 & 7 & 2 \\ 2 & 3 & 2 & 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} 1 & 2 & 3 & 3 & 5 \\ 2 & 7 & 0 & 2 & 1 \end{bmatrix}^T$.

3. Experiment Results

The experimental datasets are from several publicly attainable benchmark datasets including both the datasets in a matrix representation: the ORL face database and the Letter text-base, and the datasets in a vector representation: Wave form, Water-treatment, Wine, Sonar, Musk Clean2, Balance-scale and Monks. The dimensions of the datasets selected here are from 4 (the Balance-scale) to 166 (Musk-Cleans2) for the vector patterns and from 24x18 (Letter) to 28x23 (*face* image, down-sampled

from the original size of 112x92) for the image matrix patterns. In the experiments, each dataset was divided into two parts with fixed sizes and no overlapping - training and testing sets. For each classification problem, 10 independent runs were performed and their classification accuracies and running times on the testing sets are averaged. In Sub-section 3.1, we firstly give brief descriptions for all the datasets used here. In addition, θ is fixed to 98% to determine *d* value for all experiments carried out here.

3.1 Brief description of the Datasets

1) ORL face base (http://www.cam-orl.co.uk) contains 400 grey human face images of 40 persons, 10 different images each person, taken against a dark homogeneous background at different times and with some changes of the facial expressions. The images are in upright, frontal position with tolerance for some tilting and rotation of up to 20 degree, Moreover, the most variation of some image scale is close to 10% and the resolution of all images is normalized to 28×23. 5 selected randomly from each person are used for training and the rest for testing.
2) Letter dataset (http://sun16.cecs.missouri.edu/pgader/CECS477/NNdigits.zip) contains 10 text classes consisted of digits 0-9 with each class having 50 samples, each sample is a 24x18 matrix pattern. 25 samples drawn randomly from each class are used to training and the rest used as testing.
3) Waveform data (Blake and Merz, 1998) contains 1500 3-class vector patterns with each class having 500 and each sample having 21 dimensions. 250 chosen randomly from each class are used for training.
4) Water-treatment data (Blake and Merz, 1998) contains 116 38-dimensional 2-classes vector patterns with Class 1 and Class 2 having 65 and 51 pattern respectively. 25 selected randomly

from each class are for training and the rest for testing.

5) Wine dataset (Blake and Merz, 1998) contains 178 12-dimensional 3-class vector patterns with their three class pattern numbers being 59, 71 and 48 respectively. 24 patterns randomly from each class are for training and the rest for testing.

6) Sonar dataset (ftp://ftp.cs.cmu.edu/afs/cs/project/connect/bench/.) contains two classes (rock or cylinder), in which the cylinder class has 111 patterns and the rock class has 97, each pattern is 60 dimensions. 50 drawn randomly from each class are for training and the rest for testing.

7) Musk-Clean2 (166D) dataset (Blake and Merz, 1998) has 6598 166-dimension 2-class vector patterns, where two classes contain 1017 and 5581 patterns respectively. 500 random patterns each class are used for training and the rest for testing.

8) Musk-Clean2 (160D) dataset (Blake and Merz, 1998) is the same the dataset in 7) but the last 6 dimensions of each pattern are omitted (by us) mainly for generating more assembling matrix patterns.

9) Balance-scale (4D) dataset (Blake and Merz, 1998) has 625 3-class vector patterns, where three classes contain 49, 288 and 288 patterns respectively. 25 random patterns each class are used for training and the rest for testing.

10) Monks (6D) dataset (Blake and Merz, 1998) has 1625 2-class vector patterns, where two classes contain 723 and 902 patterns respectively. 400 random patterns each class are used for training and the rest for testing.

3.2 **Results**

The preliminary experiments conducted here aim to demonstrate applicability of both the matrixizied PCA and FLDA to vector patterns by giving different classification performances and computational

complexity on different datasets and finally additionally, confirming that MatFLDA can indeed breakdown the FLDA rank limitation.

A) Classification performance

Table 1 demonstrates all classification accuracies under the above experimental conditions. From it, we can observe that 1) for image patterns such as ORL and Letter datasets, MatPCA and MatFLDA are exactly 2DPCA (Yang, Zhang, Frangi and Yang, 2004) and the image FLDA (Liu, Cheng and Yang, 1993), respectively. Compared to PCA and FLDA, both improved consistently recognition performance on the ORL dataset by about 0.6%, respectively, and while in the Letter dataset, MatPCA accuracy is only increased by 0.4% but MatFLDA increased by 3.3%. From these facts, we believe that 2DPCA and image FLDA are indeed beneficial for image pattern classification and such a benefit seems to contribute to retaining original structural information in the image matrix and while the performance degradation of both PCA and FLDA possibly results from the vectorization for them; 2) for the vector patterns from the rest 8 datasets used here, such a performance raise does not always hold. In fact, only on 3 datasets (Water-treatment, Sonar and Monks), both improve simultaneously performance from slight to distinct under at least one matrixizing mode compared to both PCA and FLDA and whereas on another 3 datasets (Waveform, Musk-Clans2(166D) and Balance-scale), the results are exactly opposite. Finally on the remaining Wine and Musk-Cleans(160D) datasets, MatPCA obtains performance raise almost on all matrixizing modes but MatFLDA does not so. On the whole, MatPCA improves performance on 5 out of 8 datasets and especially distinct on Water-Treatment (achieving 17.15%) and while MatFLDA also behaves well on 3 out of 8 datasets, particularly on Water-Treatment and Sonar, respectively raising 7.44 and 8.41%s in performance. On the other hand, on those datasets that

their performances are degraded, MatPCA achieves 7.13% on Wine in the worst case but only slight on the rest and while MatFLDA, 3.23, 6.88 and 12.86%s, respectively on Waveform, Musk-Cleans2(166D) and Balance-Scale but only slight on the rest. Therefore, whether MatPCA and MatFLDA can really promote performance or not depends on different matrixizing modes even for the same dataset and different datasets. From different performance exhibitions for different matrixizing modes 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 mode is coincidental to nature of those data such as 2D image and Letter 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 Waveform dataset. In short, matrixization for vector patterns just gives us one more option in a lot of feature extraction and representation methods.

B) Comparison of Running times

In order to illustrate computation reduction of our two methods compared to both PCA and FLDA, we perform all these algorithms on an IBM computer with 1.7 GHz Pentium processor using Matlab (Mathworks, Inc. Natik, MA), record their running times on 8 datasets (where the minimal dimensionality is equal to or greater than 12) and only tabulate parts of both PCA and MatPCA into Table 2 but at the same time omit deliberately parts of both FLDA and MatFLDA due to similar results. From the Table 2, we can see that different matrixizations yield different computation times. Like PCA and 2DPCA, The larger the scale of S_t^{Mat}, the more the computational time particularly for large dimensional vector pattern, such speedups are distinct especially in the ORL and Letter datasets and almost reach 196 and 30 times respectively, but not at the price of sacrificing performance, while for those medium large vector pattern (here 12-166D), the computational times are basically comparable as

shown in Table 2 but still depend on the matrixizing modes, for example, for Musk2(160D) under 4 different modes, the speedup can reach about 3 times. Because the classification performances, in some cases, are indeed degraded, which also likely attributes to the matrixization. Therefore, we must seek a trade-off between performance and matrixization.

Insert Table1 and 2 here

C) The breakdown of rank limitation

As has clarified in Section 2.2, the maximal number of the projection vectors we can find in FLDA is bounded by *C*-1 (the rank of the between-scatter matrix). Hence, in the datasets used here, the ranks of their between-scatter matrices are, accordingly, at most 39, 9, 2, 1, 2, 1, 1, 2 and 1. However, MatFLDA can naturally breakdowns this limitation and need not to pay additional computational cost as long as the number of the rows or columns of a matrixized pattern is greater than *C*-1. In the datasets used here except the ORL, all the ranks of $S_b^{Mat}s$ excess their class numbers, and thus breaking down the rank limitation and obtaining more discriminant vectors. The computational results in Table 3 also confirm this: even in the case that their scatter matrices are not full ranks as underlined in the same table. The projection numbers found by MatFLDA on all the datasets except the ORL are still indeed greater than those done by FLDA. However, we must remember that the more the found discriminant projection vectors for MatFLDA, it does not imply that the better the classification performance. In fact, Table 1 also tells us such a case. In addition, what needs supplemented is for the ORL, MatFLDA still improves classification performance though the limitation is not breakdown, which still possibly contributes to obtaining a preservation of structural information in image matrix pattern. However, we can further improve its performance using 56x46 resolution of face instead of 28x23 here and thus likewise, also

break down the rank limitation. In sum, the matrixization brings us some feasibility of matrixizing a vector pattern and thus reduction on computational time to some extent but it is not necessarily always to bring performance promotion.

Insert Table 3 here

D) Remaining shortcomings

Except that the matrixization is not always yield good effectives as illustrated in A) in this subsection, Besides possible degradation of performance for some datasets due to the matrixization, there is other shortcoming as will be described. The main goal of performing all the above algorithms is to find a corresponding projection transform represented by a matrix and subsequently the extracted features by the transform are used to classification. For both PCA and FLDA, their after-projected patterns are both still vectors and their dimensionalities are far smaller than the original ones and thus so-needed memory is greatly reduced, while for MatPCA and MatFLDA, the after-projected patterns are both matrices and their sizes are generally larger than those of the former two and thus more memory is required. So, we need to achieve a balance between time-saving and memory by adjusting θ value to a suitable one, which is still investigated.

4. **Conclusions and Future work**

In this paper, we employed the concept from IMPCA, 2DPCA and image FLDA to develop MatPCA and MatFLDA, they are the corrsponding matrixizied versions for the traditional PCA and FLDA and still linear approaches, and thus still keep simple and effective. Both not only can directly analyze matrix pattern and vector pattern but also save lots of computational cost, especially for large

dimensional pattern, due to the scale reduction of all the scatter matrices concerned at the price of increasing memory for the extracted features. Our experiments on real world datasets indicate that both MatPCA and MatFLDA can consistently improve classification accuracy on image datasets used here but cannot necessarily work well for original vector patterns compared to the corresponding PCA and FLDA. And these improved or degraded extents are problem- as well as matrixization-dependent. In reaching the same or similar performance, generally MatPCA needs less projection vectors than PCA and while MatFLDA can get more optimal discriminating vectors than its counterpart. At the same time, it is noteworthy that those obtained benefits are not from popular nonlinearization techniques such as nonlinear PCA (SchÖlkopf, Smola and Muller,1998) and kernel FLDA (Mika, Ratsch, Weston, SchÖlkopf and Muller, 1999) but simply from the matrixization for the vector pattern. Perhaps, this is another cheap pavement to boost classification performance. In addition, the fact that different matrixization or assembling technique for the same vector pattern produces different performance drives us to find a more appropriate matrixization for data by using some optimization technique, which is one of our next problems to be attacked. Our future researches also include nonlinearizing MatPCA and MatFLDA by means of the recently popular kernel trick (Mika, Ratsch, Weston, SchÖlkopf and Muller, 1999) (SchÖlkopf, Smola and Muller, 1998) to further boost their performances and while not to induce too much computational cost.

Acknowledgements

We thank the anonymous reviewers' constructive comments for improving presentation of this paper and National Science Foundations of China and of Jiangsu under Grant Nos. 60473035 and BK2002092, Jiangsu natural science key project, Jiangsu "QingLan" Project Foundation and the Returnee's Foundation of China Scholarship Council for partial supports respectively.

References

Beymer, D. and Poggio, T., 1996. Image representations for visual learning. Science 272 905-1909.

Blake, C. L. and Merz, C. J., 1998. UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRository.html

Chen, S.C. and Liang, P., 2002. GIMPCA: a general technique unifying PCA and IMPCA, Tech. Report #00-10, Dept. of Computer Sci. & Eng., Nanjing Univ. of Aeronaut. & Astronaut..

Clarke, J., 1985. Transform coding of images, London: Academic Press.

Duhamael, P. and Vetterli, M., 1990. Fast Fourier transform: a tutorial review and a state of the art. Signal Processing 19 259-299.

Friedman, J. H., 1989. Exploratory projection pursuit. J. of the Amer. Stat. Asso.. 84(405) 165-175.

Fukunaga, K., 1990. Introduction to Statistical Pattern Recognition. Academic Press, San Diego, CA.

Hastie, T., Buja, A. and Tibshirani, R., 1995. Penalized discriminant analysis. Annals of Statistics 23 73-102.

Hong, Zi-Quan, 1991. Algebraic feature extraction of image for recognition. Pattern recognition 24(3) 211-219.

Liu, Ke, Cheng, Y.-Q. and Yang, J.-Y., 1993. Algebraic feature extraction for image recognition based on an optimal discriminant criterion. Pattern recognition 26(6) 903-911 1993.

Loizou, G. and Maybank, S.J., 1987. The nearest neighbor and the Bayes error rates. IEEE Trans. Patt. Anal. & Mach. Intell.. 9 254-262.

Mallat, S., 1989. A theory of multi-resolution signal decomposition: the wavelet representation, IEEE Trans. Patt. Anal. & Mach. Intell.. 11674-693.

Mclachlan G. J., 1992. Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York.

Mika, S., Ratsch, D.J., Weston, J., SchÖlkopf, B., and Muller, K. R., 1999. Fisher discriminant

analysis with kernels. In: Neural Networks for Signal Processing IX, pp. 41-48.

Okada, T. and Tomita, S., 1985. An optimal orthonormal system for discriminant analysis. Pattern Recog.. 18(2) 139-144.

SchÖlkopf, B., Smola, A., and Muller, K.-R.,1998. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10 1299-1319.

Tian, Y., Tan, T-N., Wang, Y-H and Fang, Y-C., 2003. Do singular values contain adequate information for face recognition. Pattern Recognition 36(3) 649-655.

Yang, J. and Yang, J.-Y., 2002. From image vector to matrix: a straightforward image projection technique --- IMPCA vs. PCA. Pattern Recognition 35 1997-1999.

Yang, J., Zhang, D., Frangi, A. F. and Yang, J.-Y., 2004. Two-dimensional PCA: A new approach to appearance-based face representation and recognition. IEEE Tanns. PAMI 26(1) 131-137.

Table 1: Classification Accuracy Comparison

Datasets	Classifiers	%Accuracy	
ORL	PCA	93.35	
(28x23)	MatPCA	**94.05**	
	FLDA	93.60	
	MatFLDA	**94.00**	
Letter	PCA	87.26	
(24x18)	MatPCA	**87.74**	
	FLDA	70.36	
	MatFLDA	**73.60**	
Wave-	PCA	**77.36**	
form	MatPCA	75.21(3x7)	74.98(7x3)[a]
(21D)	FLDA	**78.70**	
	MatFLDA	75.47(3x7)	76.67(7x3)
Water-	PCA	65.21	
Treat.	MatPCA	**65.30**(2x19)	**82.36**(19x2)
(38D)	FLDA	89.76	
	MatFLDA	**92.80**(2x19)	**97.20**(19x2)
Wine	PCA	74.62	
(12D)	MatPCA	**79.93**(3x4)	**77.97**(4x3)
		81.75(2x6)	73.81(6x2)
	FLDA	**94.81**	
	MatFLDA	89.08(3x4)	89.14(4x3)
		82.61(2x6)	93.05(6x2)
Sonar	PCA	80.56	
(60D)	MatPCA	79.79(6x10)	**80.69**(10x6)
	FLDA	71.13	
	MatFLDA	**78.82**(6x10)	**78.17**(10x6)
		79.54(3x20)	**73.30**(20x3)

a: Accuracy under the 7x3 dimension assembled matrix.

Table 1 (Cont'd)

Dataset	Classifier	%Accuracy	
Musk-Clean2 (166D)	PCA	80.5	
	MatPCA	80.21(2x83)	80.40(83x2)
	FLDA	86.39	
	MatFLDA	79.14(2x83)	84.14(83x2)
Musk -Clean2 (160D)	PCA	79.68	
	MatPCA	**80.09**(10x16)	**80.06**(16x10)
		80.03(8x20)	**80.16**(20x8)
	FLDA	86.57	
	MatFLDA	78.47(10x16)	79.22(16x10)
		78.86(8x20)	79.69(20x8)
Balance-Scale (4D)	PCA	62.60	
	Mat_PCA	61.53(2x2)	
	FLDA	83.47	
	Mat_FLDA	70.61(2x2)	
Monks (6D)	PCA	57.22	
	Mat_PCA	57.42(2x3)	**57.44**(3x2)
	FLDA	57.08	
	Mat_FLDA	57.16(2x3)	**57.34**(3x2)

Table 2: Running-time comparison of MatPCA and PCA

Dataset	Classifier	Time (second)	
ORL	PCA	9.120	
	MatPCA	0.0464	
Letter	PCA	2.9690	
	MatPCA	0.0372	
Wave-	PCA	0.0177	
form	MatPCA	0.0167(3x7)	0.0166 (7x3)
Water-	PCA	0.0059	
Treat.	MatPCA	0.0023 (2x19)	0.0038(19x2)
Wine	PCA	0.0047	
	MatPCA	0.0030(3x4)	0.0027(4x3)
		0.0025(2x6)	0.0030(6x2)
Sonar	PCA	0.0123	
	MatPCA	0.0063(6x10)	0.0063(10x6)
		0.0062(4x15)	0.0047(15x4)
		0.0062(3x20)	0.0062(20x3)
		0.0062(2x30)	0.0078(30x2)
Musk-	PCA	0.2845	
Clean2	MatPCA	0.1375(2x83)	0.1171(83x2)
(166D)			
Musk-	PCA	0.2719	
Clean2	MatPCA	0.0953(10x16)	
(160D)		0.0906(16x10)	
		0.0937(8x20)	0.0968(20x8)

Table 3: Ranks for scatter matrices of both FLDA and MatFLDA

Datasets	Scatter Matrix	Rank	
ORL	S_b (MatFLDA)	28	
(28x23)	S_w (MatFLDA)	28	
	S_w (FLDA)	200	
Letter	S_b (MatFLDA)	**23**	
(24x18)	S_w (MatFLDA)	**23**	
	S_w (FLDA)	250	
Wave-	S_b (MatFLDA)	3(3x7)	7(7x3)
form	S_w (MatFLDA)	3(3x7)	7(7x3)
(21D)	S_w (FLDA)	21	
Water-	S_b (MatFLDA)	2(2x19)	**4**(19x2)
Treat.	S_w (MatFLDA)	2(2x19)	19(19x2)
(38D)	S_w (FLDA)	83	
Wine	S_b (MatFLDA)	3(3x4)	4(4x3)
(12D)		2(2x6)	6(6x2)
	S_w (MatFLDA)	3(3x4)	4(4x3)
		2(2x6)	6(6x2)
	S_w (FLDA)	12	
Sonar	S_b (MatFLDA)	6(6x10)	10(10x6)
(60D)		4(4x15)	**8**(15x4)
		3(3x20)	**6**(20x3)
		2(2x30)	**4**(30x2)
	S_w (MatFLDA)	6(6x10)	10(10x6)
		4(4x15)	15(15x4)
		3(3x20)	20(20x3)
		2(2x30)	30(30x2)
	Sw (FLDA)	60	

Table 3 (Cont'd)

Dataset	Scatter Matrix	Rank	
Musk-Clean2	S_b (MatFLDA)	2(2x83)	**4**(83x2)
(166D)	S_w (MatFLDA)	2(2x83)	83(83x2)
	S_w (FLDA)	166	
Musk-Clean2	S_b (MatFLDA)	10(10x16)	16(16x10)
(160D)		8 (8x20)	**16** (20x8)
	S_w (MatFLDA)	10(10x16)	16(16x10)
		8 (8x20)	20 (20x8)
	S_w (FLDA)	160	
Balance-	S_b (MatFLDA)	2(2x2)	
Scale (4D)	S_w (MatFLDA)	2(2x2)	
	S_w (FLDA)	4	
Monks (6D)	S_b (MatFLDA)	2(2x3)	**2** (3x2)
	S_w (MatFLDA)	2(2x3)	3 (3x2)
	S_w (FLDA)	6	