Available online at www.sciencedirect.com
ScienceDirect
NEUROCOMPUTING
Neurocomputing 70 (2007) 1056–1061
www.elsevier.com/locate/neucom

Letters

A method for speeding up feature extraction based on KPCA

Yong Xu[a,*], David Zhang[b], Fengxi Song[a], Jing-Yu Yang[c], Zhong Jing[c], Miao Li[d]

[a]*Bio-Computing Research Center, Shenzhen graduate school, Harbin Institute of Technology, Shenzhen 518055, China*
[b]*The Biometrics Research Center and Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong*
[c]*Department of Computer Science & Technology, Nanjing University of Science & Technology, Nanjing, China*
[d]*College of Computer Science & Technology, Harbin Institute of Technology, Harbin, China*

Received 15 January 2006; received in revised form 6 September 2006; accepted 16 September 2006
Communicated by S. Choi
Available online 25 October 2006

Abstract

Kernel principal component analysis (KPCA) extracts features of samples with an efficiency in inverse proportion to the size of the training sample set. In this paper, we develop a novel method to improve KPCA-based feature extraction. The developed method is the first one that is methodologically consistent with KPCA. Experiments on several benchmark datasets illustrate that the feature extraction process derived from the novel method is much more efficient than that associated with KPCA. Moreover, the classification accuracy generated from the developed method is similar to that of KPCA.

Keywords: Principal component analysis (PCA); Kernel PCA (KPCA); Improved KPCA (IKPCA); Feature extraction

1. Introduction

Kernel principal component analysis (KPCA) [8,9], a widely used nonlinear feature extraction method, was derived from principal component analysis (PCA) [1–6,10,15]. For KPCA, it is certain that a feature extractor can be expanded in terms of all training samples in feature space. Thus, if we use KPCA to extract features of a sample, we should calculate all the kernel functions between this sample and the total training samples in advance and then implement feature extraction using these functions. As a result, the larger the size of the training sample set, the lower the efficiency of feature extraction. Especially, for real-world applications with large numbers of training samples, KPCA-based feature extraction will be inefficient and even unfeasible. Indeed, other kernel methods also suffer from similar problems [11–14]. Some algorithms have been proposed to accelerate feature extraction associated with kernel methods. Generally, these algorithms primarily root in the following two ideas. The first idea is based on the supposition that one or more training samples in feature space can be expressed exactly as a linear combination of the others. Another idea is that a feature extractor of the feature space may be expanded approximately in terms of some vectors, which are fewer than the total training samples and may or may not be from the training sample set.

The first idea above seems to be reasonable and feasible for linearly dependent training samples. In this case, there is at least one training sample that can be expressed exactly as a linear combination of the others. However, for some real-world applications such as the ones associated with the Gaussian kernel, the training samples in feature space cannot be linearly dependent. Most of the algorithms based on the second idea were developed only with the viewpoint of numerical approximation and it is not clear whether these algorithms are methodologically consistent with the essence of KPCA or not. In addition, it is noticeable that the expectation maximization approach proposed by Rosipal and Girolami [7] is helpful to improve the implementation efficiency of KPCA with a large number of data points, though this approach is not able to improve KPCA-based feature extraction.

With this paper, we are the first to develop such an improved KPCA method that is still subject to the KPCA

*Corresponding author. Tel./fax: +86 025 84315510.
E-mail address: laterfall2@yahoo.com.cn (Y. Xu).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.09.005

methodology. Superficially, it would appear that the idea proposed by us in this paper is somewhat formally similar to the idea for deriving fast kernel Fisher discriminant analysis (FKFDA) [12]; however, our method for improving KPCA are essentially different from that for obtaining FKFDA. Our method in this paper is derived directly from the KPCA methodology while FKFDA is not. The feature extraction process using the improved KPCA can be much more efficient than that using the original KPCA. Experimental results also show that IKPCA is similar to KPCA in classification accuracy. The rest of this paper is organized as follows. KPCA is briefly introduced in Section 2. Then the IKPCA method is presented in Section 3, followed by the experimental results shown in Section 4. Finally, the conclusion is presented in Section 5.

2. Nonlinear PCA based on a kernel function

As a nonlinear method, KPCA is nothing but the PCA in the feature space associated with a kernel function. We assume that there are N training samples, $x_1, x_2, ..., x_N$, in total. If the training samples have been mapped into a feature space by a nonlinear function ϕ, we may perform PCA based on the training samples in feature space. The correlation matrix of the feature space can be computed by $\Sigma_\phi = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)^{\mathrm{T}}$. It is easy to demonstrate that feature extractors in feature space must be from the set of the eign-vectors of Σ_ϕ. With these feature extractors, we can obtain features of samples and can also reconstruct the samples with the minimum mean-square error. Furthermore, the following equation can be derived:

$$K\alpha = \lambda\alpha, \tag{1}$$

where $(K)_{ij} = k(x_i, x_j)$. $k(x_i, x_j)$ means the kernel function between x_i and x_j. The principal component analysis method based on the eigen-equation (1) is the so-called KPCA.

It is easy to know that for the sample $\phi(x)$ in feature space, the most representative m dimensional features extracted using KPCA form the following vector:

$$Y = \left[\frac{\sum_{j=1}^{N}\alpha_j^{(1)}k(x_j,x)}{\sqrt{\lambda_1^{\alpha}}}\ \frac{\sum_{j=1}^{N}\alpha_j^{(2)}k(x_j,x)}{\sqrt{\lambda_2^{\alpha}}}\cdots\frac{\sum_{j=1}^{N}\alpha_j^{(m)}k(x_j,x)}{\sqrt{\lambda_m^{\alpha}}}\right]^{\mathrm{T}}, \tag{2}$$

where $\alpha^{(1)}, \alpha^{(2)}, ..., \alpha^{(m)}$ are, respectively, the m eign-vectors associated with the first m largest eigen-values $\lambda_1^{\alpha}, \lambda_2^{\alpha}, ..., \lambda_m^{\alpha}$ of (1). $\alpha_j^{(i)}$ denotes the jth component of the vector $\alpha^{(i)}$. According to the essence of the PCA methodology, the feature extraction procedure based on (2) is theoretically able to produce the minimum reconstruction error.

3. Idea and algorithm for improving KPCA

3.1. Idea of fast feature extraction

Section 2 has shown that, in the feature space, feature extraction can be implemented using (2). However, (2) indicates that to obtain features of a sample, we should calculate all the kernel functions between this sample and the total training samples, which means that the feature extraction process associated with a training sample set of a large size is quite inefficient. To speed up KPCA-based feature extraction, we assume that in the feature space a feature extractor can be expressed approximately as a linear combination of a portion of training samples, called nodes. The corresponding coefficients are called expansion coefficients. The assumption is supported by the fact that when a feature extractor is expanded in terms of all the training samples, different training samples have dissimilar effects on the expansion. In other words, some training samples contribute much to the expansion, whereas the others contribute less [13]. If we find out the "important" training samples, which contribute much to the expansion, and newly expand feature extractors in terms of them, then these "important" samples can be taken as the nodes. Consequently, we can extract features of a sample using all the kernel functions between this sample and the nodes. Since the nodes are fewer than the total training samples, we can lead to more efficient feature extraction process. The strategy for determining nodes will be presented in Section 3.2. Though ideas superficially similar to the above assumption have been successfully applied to kernel-based discriminant analysis methods [11,12], the corresponding methods are distinct from our method for improving KPCA presented below. One of the main differences between our method in this paper and those in previous works [11,12] is that our method is still based on the methodology of principal component analysis, whereas the others base their algorithms on the physical meaning of Fisher discriminant analysis. It is also noticeable that the method in Ref. [13] can select nodes from training samples very easily; however, the corresponding algorithm is not justified theoretically.

Suppose that a feature extractor u_i can be expanded approximately in terms of $u_i \approx \sum_{j=1}^{s}\beta_j\phi(x_j^0), s<N$; consequently, $\Sigma_\phi u_i \approx \lambda u_i$. For simplicity, we replace the sign "$\approx$" with "$=$" in the context below. Then the following set of equations is certain:

$$\left(\phi(x_k^0)\cdot\Sigma_\phi\right)u_i = \lambda\left(\phi(x_k^0)\cdot u_i\right),\ \ k = 1, 2, ..., s. \tag{3}$$

Substituting $u_i = \sum_{j=1}^{s}\beta_j\phi(x_j^0)$ into (3) arrives at

$$\lambda\sum_{j=1}^{s}\beta_j\left(\phi(x_k^0)\cdot\phi(x_j^0)\right) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{s}\beta_j\left(\phi(x_k^0)\cdot\phi(x_i)\right)\left(\phi(x_i)\cdot\phi(x_j^0)\right),\ k = 1, 2, ..., s.$$

Then, we can formulate this set of equations as follows:

$$\frac{1}{N}K_1(K_1)^{\mathrm{T}}\beta = \lambda K_2\beta, \tag{4}$$

where $\beta = [\beta_1\ \beta_2 \dots \beta_s]^{\mathrm{T}}$, $(K_1)_{ij} = k(x_i^0, x_j)$, $i = 1, 2, ..., s$, $j = 1, 2, ..., N$, $(K_2)_{ij} = k(x_i^0, x_j^0)$, $i, j = 1, 2, ..., s$.

The above demonstration relates the problem for determining nodes with an eign-value equation. In other words, for the approximate expansion of a feature extractor, we can determine the expansion coefficients based on Eq. (4). We call the principal component analysis technique using the eigen-equation (4) improved KPCA (IKPCA). Exploiting this technique, we can carry out feature extraction more efficiently, which will be shown in Section 3.2. Notice that if there are N nodes i.e. $s = N$, then (4) will be identical to the eigen-equation associated with KPCA.

3.2. Algorithm for determining nodes

According to the physical meaning of principal component analysis, the feature extractors of PCA or KPCA must be the eign-vectors of the corresponding eigen-value equation. Moreover, the performance of the feature extractors of either of the two methods can be assessed by the corresponding eigen-values. In practice, when extracting features of samples using PCA or KPCA, we prefer a feature extractor (i.e. an eign-vector) associated with a large eign–value to that associated with a small eign–value, because a large eign–value means a small construction error. IKPCA is derived from KPCA and it may be considered an approximation version of KPCA, thus the performance of a feature extractor of IKPCA can be also assessed by the corresponding eign–value generated from Eq. (4). That is, for a feature extractor (i.e. an eign–vector) from Eq. (4), the larger the eign–value associated with it is, the better it is. We propose to determine nodes using the following algorithm.

Step 1. Determine the first node.

For the ith training sample x_i, K_1, K_2, λ are computed using $K_1 = [k(x_i, x_1)\, k(x_i, x_2) \dots k(x_i, x_N)]$, $K_2 = [k(x_i, x_i)]$ and $\lambda = K_1(K_1)^{\mathrm{T}}/K_2$, respectively. Obviously, K_2 and $K_1(K_1)^{\mathrm{T}}$ are both scalars and every training sample has respective λ. When all the training samples have been searched and investigated, the one associated with the maximum λ is taken as the first node, denoted by x_1^0. Then, the matrices K_1, K_2 corresponding to x_1^0 are recorded as K_1^0, K_2^0, respectively, i.e., $K_1^0 = [k(x_1^0, x_1)\, k(x_1^0, x_2) \dots k(x_1^0, x_N)]$, $K_2^0 = [k(x_1^0, x_1^0)]$.

Step l. Determine the lth node.

If $l-1$ nodes, $x_1^0, x_2^0, ..., x_{l-1}^0$, have been determined by the previous $l-1$ steps, the lth node may be determined as follows. Firstly, a vector k_j^1 is defined as

$$k_j^1 = \left[k(x_j, x_1), k(x_j, x_2), ..., k(x_j, x_N)\right]. \tag{5}$$

Let K_1^0, K_2^0, respectively, denote the matrices K_1, K_2 based on $x_1^0, x_2^0, ..., x_{l-1}^0$, i.e., $(K_1^0)_{ij} = k(x_i^0, x_j)$, $i = 1, 2, ..., l-1$, $j = 1, 2, ..., N$; $(K_2^0)_{ij} = k(x_i^0, x_j^0)$, $i, j = 1, 2, ..., l-1$. The lth node should be from the sample set $P = \{x_j | x_j \neq x_1^0, x_2^0, ..., x_{l-1}^0\}$, which is a subset of the set of the total training samples. In this step, we will take each element of P as one candidate for the lth node and respectively assess them for selecting the optimal candidate as the lth node. When assessing a sample (i.e. an element) x_j from P, we define K_1, K_2 as

$$K_1 = \begin{bmatrix} K_1^0 \\ k_j^1 \end{bmatrix}, \quad K_2 = \begin{bmatrix} K_2^0 & (k_j^2)^{\mathrm{T}} \\ k_j^2 & k(x_j, x_j) \end{bmatrix},$$

where k_j^1 is defined as in (5), $k_j^2 = [k(x_j, x_1^0)\ k(x_j, x_2^0) \cdots k(x_j, x_{l-1}^0)]$. Using the K_1, K_2, we can construct an eign-value equation in the form of Eq. (4), and then we can work out its eign-values $\lambda_1, \lambda_2, ..., \lambda_l$. Suppose that m feature extractors are required. We introduce a variable v and define it as follows: if $l \leqslant m$, then $v = \lambda_1 + \lambda_2 + \cdots + \lambda_l$; otherwise, $v = \lambda_1 + \lambda_2 + \cdots + \lambda_m$. After all the samples (elements) in P have been researched and investigated by the above procedure, the maximum v is denoted by v_l. Then, the candidate associated with v_l, is selected as the lth node and denoted by x_l^0. K_1^0, K_2^0 are newly, respectively, defined to be the matrices K_1, K_2 based on $x_1^0, x_2^0, \dots, x_l^0$. The above procedure is not terminated until $s \geqslant N \times t$ where $t < 1$, N and s are respectively the numbers of the total training samples and the determined nodes. In practice, the s can be empirically determined. Alternatively, an additional classification procedure on the features extracted using the obtained IKPCA model can assist people to determine s (or t). That is, if s is considered to be great enough, the obtained IKPCA model can be used to extract features of training samples and then the classification result on these features can help people judge whether the current IKPCA model is powerful enough to present the sample data or not. If the classification accuracy is satisfactory, the node selection procedure can be terminated; otherwise, this procedure continues to select nodes.

After the procedure for determining nodes is terminated, the sample $\phi(x)$ in feature space can be featured by

$$f = \left[\frac{\sum_{j=1}^{s}\beta_j^{(1)}k(x_j^0, x)}{\sqrt{\lambda_1}}\ \frac{\sum_{j=1}^{s}\beta_j^{(2)}k(x_j^0, x)}{\sqrt{\lambda_2}} \cdots \frac{\sum_{j=1}^{s}\beta_j^{(m)}k(x_j^0, x)}{\sqrt{\lambda_m}}\right]^{\mathrm{T}},$$

where $\beta^{(i)} = [\beta_1^{(i)}\ \beta_2^{(i)} \dots \beta_s^{(i)}]^{\mathrm{T}}$. $\beta^{(1)}, \beta^{(2)}, \dots, \beta^{(m)}$ are the first m eign-vectors associated with the first m largest eign–values of the corresponding eign–value equation taking the form of (4), which is based on the determined nodes $x_1^0, x_2^0, \dots, x_s^0$ and all the training samples $x_1, x_2, ..., x_N$. It is clear that the computational complexities of IKPCA-based feature extraction and KPCA-based feature extraction are $o(ms)$ and $o(mN)$, respectively.

Although the expectation maximization approach to KPCA [7] can much efficiently solve the eign–value problem associated with KPCA, the consequent feature extraction process for a sample still depends on the kernel functions between this sample and all the training samples,

having the complexity of $o(mN)$. That is, this approach is not able to improve the efficiency of KPCA-based feature extraction. On the other hand, while the method proposed in this paper takes a very long time to obtain the eign–value problem associated with the improved KPCA model, it is very effective in speeding up KPCA-based feature extraction.

4. Experiments

To illustrate the efficiency and performance of IKPCA, we conduct experiments on four benchmark datasets (http://ida.first.gmd.de/~raetsch/data/). Every data set includes 100 subsets except for "Splice" which has only 20 subsets. Moreover, each subset consists of one training subset and one test subset. We use the Gaussian kernel $k(x,y)=\exp(-||x-y||^2/2\sigma^2)$, and let σ^2 be equal to the square of Frobenius norm of the correlation matrix of the first training subset. The training procedure is performed on the first training subset, and then test is implemented for all the test subsets using the nearest neighbor classifier. For each dataset, we test IKPCA with different t as shown in Table 2.

Every test subset has a classification error rate, so we can figure out the average error rate of one dataset. We can also obtain the deviation of the classification error rate on a dataset. Tables 1 and 2 show the experimental results of KPCA and IKPCA on the four data sets. It appears that IKPCA extracts features of samples much more efficiently than KPCA. It is noticeable that, for the data set "Splice" whose data dimensionality is 60, IKPCA classifies much more accurately than KPCA. For the data set "Banana", the classification accuracy of IKPCA is slightly lower than that of KPCA. As for the classification performance on data sets "Diabetes" and "Cancer", there are similar classification error rates for the two methods when t is not less than 0.5. For the cases in which t is less than 0.5, IKPCA generally obtained slightly higher error rates for these two datasets. To further assess IKPCA, we obtain an IKPCA model using one training sample subset and then classify all the other training sample subsets and the test samples using the features generated from every IKPCA model, respectively. The average error rate and the deviation presented in Table 3 also show that IKPCA is capable of obtaining the most representative features of samples. All these results indicate that IKPCA does perform well in feature extraction as we expect.

5. Conclusion

If KPCA is used to extract features of a sample, all the kernel functions between this sample and the total training samples should be computed in advance. As a result, for real-world applications with large numbers of training samples, KPCA will perform feature extraction much inefficiently. Although some methods have been proposed to improve KPCA for achieving efficient feature extraction, they all do not take the principle of KPCA into account in establishing the improved KPCA model. In this paper, we develop the IKPCA algorithm to improve KPCA for more efficient feature extraction. The algorithm is feasible and reasonable; besides it is still subject to the PCA methodology, which makes it distinct from the existing algorithms for improving KPCA. For IKPCA, the feature extractors also should be the eign–vectors associated with large eign–values of the corresponding eign–value equation. The experimental results on the benchmarks show that IKPCA-based feature extraction is much more efficient than

Table 1
Experimental result of KPCA on benchmark data sets

	Number of feature extractors	Average error rate and the deviation of the error rate	Total number of training samples	Feature extraction time (s)	Training time (s)
Splice	100	25.5(2.6)	1000	2130	56.0
	90	24.7(2.5)		2096	55.6
	80	24.0(2.4)		2036	55.1
	70	21.8(2.2)		2020	54.6
Diabetes	100	11.5(2.8)	468	380	7.9
	90	11.7(2.8)		360	7.1
	80	11.5(2.8)		350	7.0
	70	11.8(2.9)		341	6.7
Banana	100	13.8(0.2)	400	8147	5.2
	90	13.8(0.2)		7875	4.9
	80	13.8(0.2)		7640	4.6
	70	13.8(0.2)		7059	4.4
Cancer	70	9.0(3.2)	200	27.6	0.99
	60	8.5(3.0)		25.9	0.94
	50	8.5(3.0)		25.3	0.92
	40	9.8(3.3)		23.8	0.84

Notice that for two numbers A,B in the form of A(B) in this table, A means the average error rate and B denotes the deviation of the error rate.

Table 2
Experimental result of IPCA on benchmark data sets

	Number of feature extractors	Average error rate and the deviation of the error rate				Feature extraction time (s)				Training time (s)			
Splice		$t=0.1$	$t=0.15$	$t=0.2$	$t=0.25$	$t=0.1$	$t=0.15$	$t=0.2$	$t=0.25$	$t=0.1$	$t=0.15$	$t=0.2$	$t=0.25$
	100	18.6(1.8)	18.8(1.9)	18.4(1.8)	17.8(1.8)	799	802	808	813	1254	4225	8433	15675
	90	18.2(1.8)	17.8(1.8)	18.0(1.8)	17.8(1.8)	778	782	785	789	1240	4148	8412	15584
	80	18.7(1.9)	17.3(1.7)	18.4(1.8)	17.5(1.8)	762	763	767	781	1236	4114	8395	15541
	70	19.1(1.9)	17.2(1.7)	17.7(1.8)	17.6(1.7)	735	739	743	749	1231	4045	8371	15529
Diabetes		$t=0.3$	$t=0.4$	$t=0.5$	$t=0.6$	$t=0.3$	$t=0.4$	$t=0.5$	$t=0.6$	$t=0.3$	$t=0.4$	$t=0.5$	$t=0.6$
	100	11.7(2.9)	11.7(2.8)	11.4(2.8)	11.4(2.8)	187	212	234	257	1217	3064	6395	10232
	90	12.1(2.9)	12.2(2.9)	11.9(2.9)	12.1(2.9)	186	209	230	251	1208	3055	6374	9819
	80	11.4(2.7)	11.6(2.8)	11.6(2.8)	11.9(2.9)	181	203	227	248	1204	3053	6335	9620
	70	12.4(3.0)	12.3(2.9)	12.0(2.9)	12.1(2.0)	173	199	221	242	1199	3051	6329	9561
Banana		$t=0.3$	$t=0.4$	$t=0.5$	$t=0.6$	$t=0.3$	$t=0.4$	$t=0.5$	$t=0.6$	$t=0.3$	$t=0.4$	$t=0.5$	t = 0.6
	100	14.7(0.2)	14.2(0.2)	14.1(0.2)	14.2(0.2)	2809	3771	5038	6701	527	1272	3519	6112
	90	14.1(0.2)	14.2(0.2)	14.2(0.2)	14.1(0.2)	2769	3623	4996	6395	505	1264	3291	5009
	80	14.2(0.2)	14.2(0.2)	14.2(0.2)	14.1(0.2)	2599	3544	4584	5822	484	1262	3194	4987
	70	14.3(0.3)	14.2(0.2)	14.2(0.2)	14.2(0.2)	2573	3405	4328	5626	454	1259	2740	4971
Cancer		$t=0.4$	$t=0.45$	$t=0.5$	$t=0.55$	$t=0.4$	$t=0.45$	$t=0.5$	$t=0.55$	$t=0.4$	$t=0.45$	$t=0.5$	t = 0.55
	70	9.9(3.5)	9.6(3.4)	9.2(3.3)	9.3(3.2)	11.5	12.2	13.5	13.6	80.8	131.5	154.4	210.8
	60	9.2(3.2)	8.9(3.1)	8.6(2.9)	8.9(3.0)	10.8	11.4	12.1	12.8	79.1	113.8	154.3	209.9
	50	8.5(3.0)	8.2(2.9)	8.6(2.9)	8.2(2.9)	10.4	11.1	11.7	12.4	77.9	111.7	153.3	209.8
	40	8.1(3.0)	7.9 (2.9)	8.1(2.9)	8.1(2.9)	9.8	10.6	11.2	11.8	77.7	111.1	153.0	209.5

Notice that for two numbers A,B in the form of A(B) in this table, A means the average error rate and B denotes the deviation of the error rate.

Table 3
Experimental results (the means of the average error rate and the deviation) of KPCA and IKPCA models generated from every training sample subset

	Splice	Diabetes	Banana	Cancer
KPCA	22.1(2.0)	12.2(1.5)	14.4(1.6)	10.6(1.9)
IKPCA	18.0(2.3)	12.1(1.7)	14.5(0.7)	10.4(1.2)

The number of features extractors used for feature extraction is 70. For data set "Splice" t is set to be 0.25, while for the other data sets t is set to be 0.5. The number in the bracket means the mean of the deviation of the corresponding error rates.

KPCA-based feature extraction. Moreover, classification using the features generated from IKPCA can produce a satisfactory accuracy. On the other hand, IKPCA obtains the efficient feature extraction process at the extra cost of running the time-consuming node selection procedure.

Acknowledgments

This article is partly supported by National Natural Science Foundation of China (Nos. 60602038, 60472060 and 60473039) and Natural Science Foundation of Guangdong Province, China (No. 06300862).

References

[1] Z. Bian, X. Zhang, Pattern Recognition (in Chinese), Tsinghua University Press, Beijing, 2000.
[2] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, China Machine Press, Beijing, 2004.
[3] K. Fukunaga, Introduction to Statistical Pattern Recognition, second ed, Academic Press, Inc., New York, 1990.
[4] Z. Jin, F. Davoine, Z. Lou, J.-Y. Yang, A novel PCA-based Bayes classifier and face analysis, in: IAPR International Conference on Biometrics (ICB2006), Hong Kong, January 5–7, 2006.
[5] M. Kirby, L. Sirovich, Application of the KL procedure for the characterization of human faces, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1) (1990) 103–108.
[6] C. Liu, Gabor-based kernel PCA with fractional power polynomial models for face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 26 (5) (2004) 572–581.
[7] R. Rosipal, M. Girolami, An expectation maximization approach to nonlinear component analysis, Neural Comput. 13 (2001) 505–510.
[8] B. Schölkopf, A. Smola, K.R. Müller, Kernel principal component analysis, Artificial Neural Networks-ICANN'97, Berlin, 1997, pp. 583–588.
[9] B. Schölkopf, A. Smola, K.R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[10] M. Turk, A. Pentland, Face recognition using eigenfaces, in: Proceedings of the IEEE Conference On Computer Vision and Pattern Recognition, 1991, pp. 586–591.
[11] Y. Xu, J.-Y. Yang, J. Lu, D.-J. Yu, An efficient renovation on kernel Fisher discriminant analysis and face recognition experiments, Pattern Recogn. 37 (2004) 2091–2094.
[12] Y. Xu, J.-Y. Yang, J. Yang, A reformative kernel Fisher discriminant analysis, Pattern Recogn. 37 (2004) 1299–1302.
[13] Y. Xu, J.-Y. Yang, J.-F. Lu, An efficient kernel-based nonlinear regression method for two-class classification, in: Proceedings of the

2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, August, 2005, pp. 4442–4445.
[14] Y. Xu, D. Zhang, Z. Jin, M. Li, J.-Y. Yang, A fast kernel-based nonlinear discriminant analysis for multi-class problems, Pattern Recogn. 39 (6) (2006) 1026–1033.
[15] J. Yang, J.-Y. Yang, Why can LDA be performed in PCA transformed space?, Pattern Recogn. 36 (2) (2003) 563–566.

Yong Xu received his B.S. degree and the M.S. degree in 1994 and 1997, respectively. He received his Ph.D. degree in pattern recognition and intelligence system from NUST in 2005. Now he works as a postdoctoral research fellow at Shenzhen graduate school, Harbin Institute of Technology. His current interests include face recognition, handwritten character recognition, linear and nonlinear feature extraction methods.

Fengxi Song was born in Anhui, China, on February 29, 1964. He received his B.S. degree in Mathematics at Anhui University, China, in 1984. He received his M.S. degree in Applied Mathematics at Changsha Institute of Technology, China, in 1987 and the Ph.D. degree in Pattern Recognition and Intelligence Systems, at Nanjing University of Science and Technology, China, in 2004. Now, he is a professor at New Star Research Inst. of Applied Tech. in Hefei City, China, and a postdoctoral research fellow in Bio-Computing Research Center, Shenzhen Graduate School, Harbin Institute of Technology, China. His research interests include computer vision, machine learning, and automatic text categorization.

David Zhang graduated in Computer Science from Peking University. He received his M.Sc. in Computer Science in 1982 and his Ph.D. in 1985 from the Harbin Institute of Technology (HIT). From 1986 to 1988 he was a Postdoctoral Fellow at Tsinghua University and then an Associate Professor at the Academia Sinica, Beijing. In 1994 he received his second PhD in Electrical and Computer Engineering from the University of Waterloo, Ontario, Canada. Currently, he is a Chair Professor at the Hong Kong Polytechnic University where he is the Founding Director of the Biometrics Technology Centre UGC/CRC) supported by the Hong Kong SAR Government. He also serves as Adjunct Professor in Tsinghua University, Shanghai Jiao Tong University, Beihang University, Harbin Institute of Technology, and the University of Waterloo. He is the Founder and Editor-in-Chief, International Journal of Image and Graphics (IJIG); Book Editor, Springer International Series on Biometrics (KISB); Organizer, the International Conference on Biometrics Authentication (ICBA); Associate Editor of more than ten international journals including IEEE Trans on SMC-A/SMC-C/Pattern Recognition; Technical Committee Chair of IEEE CIS and the author of more than 10 books and 160 journal papers. Professor Zhang is a Croucher Senior Research Fellow, Distinguished Speaker of the IEEE Computer Society, and a Fellow of the International Association of Pattern Recognition (IAPR).

Jing-Yu Yang received his B.S. degree in Computer Science 2

from NUST, Nanjing, China. From 1982 to 1984 he was a visiting scientist at the 3 Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. From 4 1993 to 1994 he was a visiting professor at the Department of Computer Science, 5 Missuria University. In 1998, he acted as a visiting professor at Concordia 6 University in Canada. His current research interests are in the areas of pattern 7 recognition, image processing and artificial intelligence, and expert system.

Zhong Jin is a Professor at School of Computer Science and Technology, Nanjing University of Science and Technology (NUST), China. He received his Ph.D. degree in pattern recognition and intelligence system from NUST in 1999. He visited Department of Computer Science and Engineering, the Chinese University of Hong Kong in 2000-2001 and visited Labortoire CNRS HEUDIASYC, Université de Technologie de Compiègne, France in 2001–2002. He visited Centre de Visió per Computador, Universitat Autònoma de Barcelona, Spain as a Ramón y Cajal program Research Fellow from September 2003 to January 2006. His current interests are in areas of pattern recognition, image analysis, machine learning, computer vision, face recognition, facial expression analysis, and content-based image retrieval.

Miao Li received the B.S. degree and the M.S. degree 2003 and 2006, respectively. Now she is a Ph.D. student in Harbin Institute of Technology, Harbin, China. Her current interests include pattern recognition, image processing, neural network, and speech recognition.