PCA vs. ICA for Face Recognition

Oyoung LEE †, Hyeyoung PARK §, Seungjin CHOI †

† Department of Electrical Engineering, Chungbuk National University, KOREA
Tel: +82-431-261-2421, Fax: +82-431-263-2419
E-mail: {loy250, schoi}@engine.chungbuk.ac.kr

§ Department of Computer Science Yonsei University, KOREA
Tel: +82-2-365-4598, Fax: +82-2-365-2579
Email: hypark@csai.yonsei.ac.kr

Abstract: The information-theoretic approach to face recognition is based on the compact coding where face images are decomposed into a small set of basis images. Most popular method for the compact coding may be the principal component analysis (PCA) which eigenface methods are based on. PCA based methods exploit only second-order statistical structure of the data, so higher-order statistical dependencies among pixels are not considered. Independent component analysis (ICA) is a signal processing technique whose goal is to express a set of random variables as linear combinations of statistically independent component variables. ICA exploits high-order statistical structure of the data that contains important information. In this paper we employ the ICA for the efficient feature extraction from face images and show that ICA outperforms the PCA in the task of face recognition. Experimental results using a simple nearest classifier and multi layer perceptron (MLP) are presented to illustrate the performance of the proposed method.

Keywords: Eigenface, face recognition, ICA, PCA.

1. Introduction

Face recognition is a complex and difficult problem that is important for surveillance and security, telecommunications, and human-computer intelligent interactions. The task of face recognition is to recognize a person in the scene using a stored database of faces, given still or video images of the scene. Most popular statistical methods for face recognition are eigenface methods that are based on PCA [7,3].

The PCA finds an orthogonal projection that captures the maximal retained variances of input data. PCA based methods exploit only second-order structure of the data, so higher-order statistical dependencies among pixels are not considered.

Independent component analysis (ICA) is a signal processing technique which aims at finding a linear transformation to variables that are maximally statistically independent. Thus higher-order statistical structure is incorporated.

In this paper, we apply the ICA to the task of face recognition and find a statistically independent feature in the reduced feature space that is already found by PCA. We compare the performance of PCA and ICA using the AR database [5]. A simple nearest classifier and MLP are used for classification. Experimental results show that the ICA outperforms the PCA in the task of face recognition.

2. Feature Extraction

I. Principal Component Analysis (PCA)

Each of the pixel values in a sample image is considered as a coordinate in a high dimensional space, i.e., the image space. Let us consider a set of N sample face images, $\{\boldsymbol{x}_1,...,\boldsymbol{x}_N\}$, each of which belongs to m-dimensional image space. In general, face images are very high dimensional space, so a dimensionality reduction scheme is required. The PCA is known as an efficient tool for dimensionality reduction.

Let us consider a linear transformation from m-dimensional image space to n-dimensional feature space $(n < m)$. The feature vectors $\{z_i\}$ are defined by

$$z_i = \boldsymbol{U}\boldsymbol{x}_i, \tag{1}$$

where $\boldsymbol{U} \in \mathbf{R}^{n\times m}$ is a linear transformation matrix with orthonormal rows. In PCA, the matrix $\boldsymbol{U}$ is chosen to minimize the reconstruction error. It is known that the rows of $\boldsymbol{U}$ correspond to the principal eigenvectors of the sample covariance matrix of $\{\boldsymbol{x}_i\}$. It is briefly explained below.

The average face $\boldsymbol{\mu}$, is defined by

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i. \tag{2}$$

Then, the sample covariance matrix $\boldsymbol{C}_x$ is given by

$$\begin{aligned}\boldsymbol{C}_x &= \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T \\ &= \frac{1}{\mathrm{N}}\boldsymbol{\Phi}\boldsymbol{\Phi}^T,\end{aligned} \tag{3}$$

where $\boldsymbol{\Phi}$ is

$$\boldsymbol{\Phi} = [\boldsymbol{x}_1-\boldsymbol{\mu},...,\boldsymbol{x}_N-\boldsymbol{\mu}]. \tag{4}$$

Since the sample covariance matrix $\boldsymbol{C}_x$ is symmetric, we have the following eigen-decomposition:

$$\boldsymbol{C}_x = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T, \tag{5}$$

where $\boldsymbol{V}$ is the modal matrix (the column vectors of $\boldsymbol{V}$ correspond to the normalized eigenvectors of $\boldsymbol{C}_x$) and $\boldsymbol{\Lambda}$ is the corresponding eigenvalue matrix whose diagonal elements are arranged in descending order of their magnitude. First n column vectors are selected and are assigned to the n row vectors of the matrix $\boldsymbol{U}$ in PCA. The n row vectors of the matrix $\boldsymbol{U}$ are called eigenfaces and are served as basis images.

Snap-shot method

If the number of sample images, N is less than the dimension of the image space, m, then it was argued in

[7] that the m-dimensional eigenvectors of C_x can be computed by first finding the eigenvectors of the matrix $\Phi^T\Phi$. Let us consider the eigenvectors $\{e_i\}$ of the matrix $\Phi^T\Phi$ such that

$$\Phi^T\Phi e_i = \lambda_i e_i . \tag{6}$$

Premultiplying both sides by Φ, we have

$$\Phi\Phi^T\Phi e_i = \lambda_i \Phi e_i . \tag{7}$$

One can see that Φe_i are the eigenvectors $\Phi\Phi^T$. This approach reduces the computational complexity. But these eigenvectors are not orthogonal, so we need orthogonalization technique. For example, Gram-Schmidt orthogonalization method can be employed.

II. Independent Component Analysis (ICA)

In signal transformation or data representation, it is often useful to represent the data as a linear superposition of basis vectors. In complete representation, the n-dimensional data $z = [z_1,...,z_n]^T$ is written as

$$z = \sum_{i=1}^{n} s_i a_i , \tag{8}$$

where $\{a_i\}$ is a set of basis vectors that spans the n-dimensional vector space and $\{s_i\}$ is a set of basis coefficients that represents the activity of the corresponding basis vectors.

Many statistical models are generative models that make use of latent variables to describe probability distributions over observations. Let us define the matrix $A = [a_1,...,a_n]$ and the vector $s = [s_1,...,s_n]^T$. Then Eq. (8) can be written as

$$z = As . \tag{9}$$

In fact that model (9) is a linear generative model in the limit of zero noise. The goal of ICA is to find both A and s given only z, under the assumption of statistical independence of $\{s_i\}$ and A is a full rank matrix.

In the framework of latent variable model, basis coefficients $\{s_i\}$ can be viewed as latent variables that are not directly observable to us, but are observed through the data z. The matrix A represents a linear transformation from latent space to data space.

Learning basis vectors can be achieved by maximizing the probability of data given model. For a set of N independent data vectors $\{z_i\}_{i=1}^N$, the likelihood function is given by

$$\prod_{i=1}^{N} p(z_i \mid A). \tag{10}$$

A single factor in the log-likelihood function has the form

$$\log p(z \mid A) = -\log|\det A| + \log p(A^{-1}z). \tag{11}$$

Note that $p(A^{-1}z)$ is factored into the product of marginal density functions due to the assumption of statistical independence of basis coefficients.

Let us define

$$y = A^{-1}z. \tag{12}$$

Then the vector y is the estimate of basis coefficient vector s. In the complete representation, the vector y is easily computed through (12) after we estimate the matrix A by maximizing the log likelihood (11).

The learning algorithm for updating A can be derived using the natural gradient that is shown to be efficient in on-line learning [1]. Taking the statistical independence among $\{y_i\}$ into account, the log likelihood is given by

$$\log p(z \mid A) = -\log|\det A| + \sum_{i=1}^{n} \log p_i(y_i), \tag{13}$$

where $\{p_i(\cdot)\}$ are probability density functions of basis coefficients $\{s_i\}$. One can easily see that different priors for basis coefficients result in different log-likelihood functions.

Let us define a function

$$\varphi_i(y_i) = -\frac{d\log p_i(y_i)}{dy_i}. \tag{14}$$

with this definition, we calculate an infinitesimal increment of the second term in (13),

$$d\left\{\sum_{i=1}^{n} \log p_i(y_i)\right\} = -\varphi^T(y)dy , \tag{15}$$

where

$$\varphi(y) = [\varphi_1(y_1),...,\varphi_n(y_n)]^T , \tag{16}$$

We define a modified differential matrix dB as

$$dB = A^{-1}dA . \tag{17}$$

with this definition, we have

$$d\left\{\sum_{i=1}^{n} \log p_i(y_i)\right\} = \varphi^T(y)dBy , \tag{18}$$

and

$$d\{\log|\det A|\} = \mathrm{tr}\{dB\}, \tag{19}$$

where $\mathrm{tr}\{\cdot\}$ is the trace operator.

Combining (18) and (19), the infinitesimal increment of the log-likelihood (13) is given by

$$d\{\log p(z \mid A)\} = -\mathrm{tr}\{dB\} + \varphi^T(y)dBy . \tag{20}$$

The differential in (20) is in terms of the modified differential matrix dB. Since dB is a linear transformation of dA, dB represents a valid search direction to maximize (13) as long as dA is nonsingular, because dB spans the same tangent space of matrices as spanned by dA.

We calculate the derivative of the log-likelihood with respect to the modified differential matrix dB. Invoking (20), we have

$$\frac{d\{\log p(z \mid A)\}}{dB} = -I + \varphi(y)y^T . \tag{21}$$

Gradient ascent method leads to the updating rule for B in the following form

$$\begin{aligned}\Delta B &= \eta_t \frac{d\{\log p(z \mid A)\}}{dB} \\ &= -\eta_t\{I - \varphi(y)y^T\}.\end{aligned} \tag{22}$$

where $\eta_t > 0$ is a learning rate and ΔB represents the difference between the current value of B and the previous value of B.

From the relationship between the modified differential matrix dB and the differential matrix dA, we have the following learning rule for updating A,

$$\Delta A = -\eta_t A\{I - \varphi(y)y^T\}. \quad (23)$$

At each iteration, the estimate of basis coefficient basis vector, y is computed by $y = A^{-1}z$ using the current estimate of A. Then the value of A is updated by (23). This procedure is repeated until A converges.

Alternatively it is possible to learn A^{-1} instead of A. The A^{-1} coincides with the ICA filter [4,6]. Let us define $W = A^{-1}$. In similar manner, the maximization of the log-likelihood leads to the learning algorithm for updating W in the following form.

$$\Delta W = \eta_t\{I - \varphi(y)y^T\}W. \quad (24)$$

3. Classification

I. Nearest Classification

The simplest method for determining which face class provides the best description of an input face image is to find the face class k that minimizes the Euclidian distance

$$\varepsilon_k = \|\Omega - \Omega_k\|^2 \quad (25)$$

where Ω is a feature vector of input face image (test data set). And Ω_k is a feature vector of training data set which describing the k th face class.

II. MLP Classification

We use a feedforward neural network in order to increase the performance of the classification. A two-layer feedforward neural network with M hidden nodes and L output nodes can be defined by deterministic functions of the forms

$$f_i(z;\theta) = \sigma_O\left(\sum_j^M v_{ij}\sigma_h(w_j \cdot z + b_j) + c_i\right), \quad i = 1,...,L \quad (26)$$

where each $f_i(z;\theta)$ corresponds to the i th output node, respectively. The parameter vector θ includes all parameters w_j, v_{ij}, b_j, and c_i. The activation function $\sigma_h(\cdot)$ for hidden nodes is a sigmoid function, and the activation function $\sigma_O(\cdot)$ for output nodes is defined by

$$\sigma_O(net_i) = \frac{\exp(net_i)}{\sum_{k=1}^{L}\exp(net_k)} \quad (27)$$

$$net_i = \sum_j^M v_{ij}\sigma_h(w_j \cdot z + b_j) + c_i \quad (28)$$

in order that the summation of the values of all output nodes becomes unity.

We use this neural network model for estimating the conditional probability of random vector y given input feature z. The corresponding conditional probability density function can be described by

$$p(y \mid z;\theta) = \prod_i^L f_i(z;\theta)^{y_i}. \quad (29)$$

Here the random vector y represents which class the input z is given from, and it is defined as

$$y_i = \begin{cases} 1 & i = \arg\min_i f_i(z;\theta) \\ 0 & \text{otherwise.} \end{cases} \quad (30)$$

The stochastic gradient descent learning method is used to find an optimal value of the parameter θ, which maximizes the log likelihood function of the form

$$l(z, y^*;\theta) = \sum_{i=1}^{L} y_i^* \log f_i(z;\theta). \quad (31)$$

The update rule at each learning step t can be written as

$$\theta_{t+1} = \theta_t + \eta_t \frac{\partial l(z_t, y_t^*;\theta_t)}{\partial \theta_t}. \quad (32)$$

where η_t is a learning rate.

4. Simulation Results

We evaluated face recognition performance for the PCA method and the ICA method using the AR face database [5]. The data set contains frontal view faces of 40 people with different facial expressions (neutral expression, smile, anger) and illumination conditions (left-light-on, right-light-on) from two sessions which are separated by two weeks. There were 400 face images (40 people * 5 images * 2 sessions), 200 face images of which were selected as the training set (neutral expression, anger, and right-light-on from first session; smile and left-light-on from second session). The rest of face images (200 face images) were chosen as the test set (smile and left-light-on from first session; neutral expression, anger, and right-light-on from second session). Each face image was cropped to the size of 46×50 and the rows of face images were concatenated to produce 2300×1 column vectors. Sample images for the training set and the test set are shown in Figures 1 and 2.

Figure 1: Sample images in the training set.

Figure 2: Sample images in the test set.

In order to find the eigenvectors of the sample covariance matrix of the data, we tested the following 3 methods.

Method 1. We compute the full covariance matrix $(m \times m)$.

Method 2. Snap-shot method.

Method 3. Snap-shot method followed by Garm-schmidt orthogonalization.

First twenty eigenfaces (which correspond to row vectors of U) are shown in Figure 3. and twenty ICAfaces are shown in Figure 4. which correspond to column vector of A. The classification was performed using a nearest neighbor classifier. The results are in Figure 5. The classification using neural networks, we used 40 hidden nodes. The learning rate was empirically selected to get

good convergence. We stopped the learning when the mean squared error of the network output becomes smaller than 0.001. The results are in Figure 6.

5. Conclusions

In the information-theoretic approach to face recognition, face images are decomposed into a set of basis images and feature vectors are obtained by projecting face images into basis images. In this paper, we have applied the ICA method to obtain basis images. In contrast to the PCA where only second-order structure of the data was used, the ICA exploited higher-order statistical structure of the data which may contain much of important information. We have shown that the ICA outperformed the PCA in the task of face recognition.

6. Acknowledgment

This research was supported by Brain Science and Engineering Research Program sponsored by Korean Ministry of Science and Thchnology.

7. References

[1] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251-276, Feb. 1998.

[2] M. S. Bartlett, H. M. Lades, and T. J. Sejnowski. Independent component representations for face recognition. In *Proceedings of the SPIE Symposium on Electronic Imaging: Science and Technology; Conference on Human Vision and Electronic Imaging III*, pages 528-539, San Jose, California, Jan. 1998.

[3] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):711-720, Oct. 1997.

[4] A. Bell and T. Sejnowski. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129-1159, 1995.

[5] A. Martinez and R. Benavente. The AR face database. Technical Report CVC #24, Computer Vision Center, Purdue University, June 1998.

[6] B. A. Olshausen and D. J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607-609, 1996.

[7] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71-86, 1991.

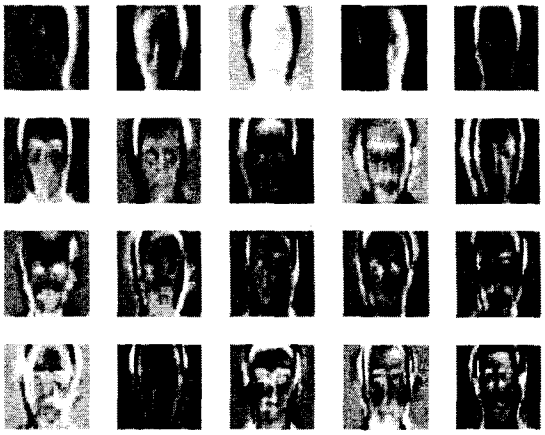

Figure 3. First 20 eigenfaces, ordered by coluymn, then, by row.

Figure 4. 20 ICAfaces, ordered by coluymn, then, by row

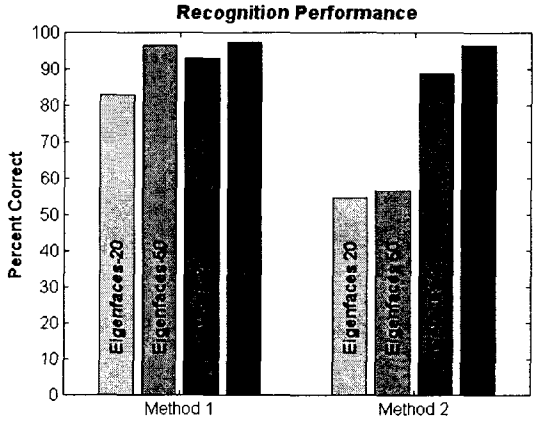

Figure 5. The comparison of recognition performance

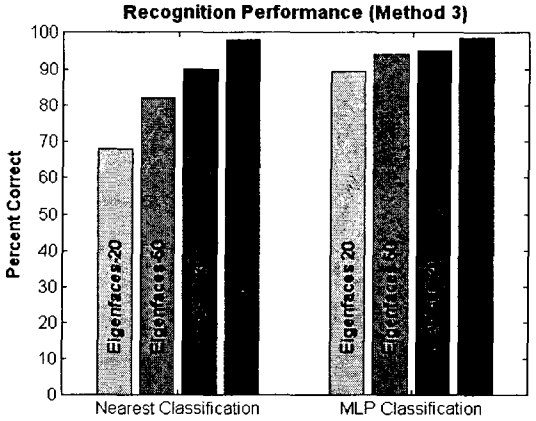

Figure 6. The comparison of recognition performance between nearest classifier and MLP classifier