Principal Component Analysis for Data Compression and Face Recognition

DINESH KUMAR[1]
C S RAI[2]
SHAKTI KUMAR[3]

[1]Department of CSE
Guru Jambheshwar University of Science and Technology
Hisar, Haryana, INDIA
[1]dinesh_chutani@yahoo.com
[2]University School of Information Technology
GGS Indraprashtha University, Delhi, INDIA
[2]csrai_ipu@yahoo.com
[3]Institute of Science and Technology Klawad
Yamuna Nagar, Haryana, INDIA
[3]shakti@istk.org

Abstract. Data compression is the most important step in many signal processing and pattern recognition applications. We come across very high dimensional data in such applications. Before processing of large-dimensional datasets, we need to reduce the dimensions to have lesser storage space and reduced computational complexities while retaining the maximum information. Principal Component Analysis (PCA) is one such technique that helps in reduction of high dimensional data. It is an unsupervised, useful statistical technique that has been successfully used in dimensionality reduction in pattern recognition applications. There are number of ways of performing Principal Component Analysis. This paper reviews the performance of three such methods, Eigen Decomposition, Singular Value Decomposition and Hebbian Neural Networks. It shows the application of the methods for face images for compression/dimensionality reduction and face recognition.

Keywords: Principal Component Analysis, Eigen Decomposition, Singular Value Decomposition, Hebbian Neural Networks, Data Compression, Face Recognition.

(Received June 17, 2008 / Accepted September 29, 2008)

1 Introduction

Principal Components Analysis (PCA) is a statistical technique that linearly transforms an original set of variables into substantially smaller set of uncorrelated variables that represent most of the information contained in the original set of variables. It transforms the data to a new coordinate system in such a way that the largest variance by any projection of the data comes to lie on the first coordinate. This first coordinate is known as first principal component, the second largest variance on the second coordinate, and so on. These principal components are orthogonal. PCA is theoretically the optimum linear transform for a given data in least mean square errors. It is used to reduce the dimensionality of the original high dimensional data set by compressing them into lower dimensions and reconstructing the original data. A small set of uncorrelated variables is much easier to understand and use in further analysis than a larger set of correlated variables. The idea was originally conceived by Pearson and independently developed by Hotelling [1, 2, 4, 17]. In some applications it is also known as Karhunen-Loéve (KL) Transform or Hotelling Transform.

This technique has been used successfully for many image processing and pattern recognition applications [16,7]. In such applications we choose the features that are often correlated with each other and a number of them are of no use so far as the discriminability is concerned. We can reduce the number of features that would require lesser storage and computational complexity is also reduced. Eigen Decomposition (ED) is one such method for computing the principal component analysis. PCA has also been developed based on the matrix theory for Singular Value Decomposition (SVD) [6, 9] and using neural networks [12, 13, 14]. This paper reviews performance of PCA using ED, SVD and Neural Networks for two different applications. These applications are data compression/dimensionality reduction and face recognition. The remainder of this paper is organized as follows. Section 2 presents basics of PCA. Section 3 explains data compression/dimensionality reduction methods followed by the description of how these methods are used for compression/dimension reduction and face recognition applications in section 4. Section 5 shows the experimental results and discussion for data compression and face recognition and paper is finally concluded in section 6.

2 Principal Component Analysis

In many real world problems, reducing dimensionality of a problem is an essential step before any analysis of data is performed. The general criterion for reducing the dimensions is the need to preserve most of the relevant information of the original data according to some optimality criteria. PCA [15] is concerned with explaining the variance-covariance structure of a set of variables through linear combinations of these variables.

Consider a random vector $X = (x_1, x_2, \ldots, x_p)$. The covariance matrix of X is C and the eigenvalues are $\lambda_1, \lambda_2, \ldots, \lambda_p$ such that $\lambda_1 \geq \lambda_2 \geq, \cdots, \geq \lambda_p$ and the eigenvector-eigenvalue pairs are $(V_1, \lambda_1), (V_2, \lambda_2), \cdots, (V_p, \lambda_p)$. Then Principal Components (PCs) are linear combinations $(y_1, y_2, \ldots, y_p)$, with the changed coordinate system of p random variables $(x_1, x_2, \ldots, x_p)$. The first principal component, y_1 is a linear combination of $x_1, x_2, \ldots, x_p$ that is

$$y_1 = (b_{11}x_1 + \cdots + b_{1p}x_p) = \sum_{i=1}^{p} b_{1i}x_i = b_1^T X \quad (1)$$

The first principal component y_1 is such that its variance is maximized given the constraint that $b_1^T b_1 = 1$. Principal components analysis finds the optimal weight vector $(b_{11}, b_{12}, ..., b_{1p})$ and associated variance of y_1 which is usually denoted as λ_1. The second principal component involves finding a second weight vector $(b_{21}, b_{22}, ..., b_{2p})$ such that the variance of y_2 is maximized subject to the constraints that $b_2^T b_2 = 1$ and the associated variance value is denoted by λ_2. This process can be continued until as many components as variables have been calculated. The sum of variance of principal components is equal to the sum of the variance of original variables such that $\sum_{i=1}^{p} \lambda_i = \sum_{i=1}^{p} \sigma_i^2$ where λ_i is the variance of the ith principal component. This way there are p linear transformations (PCs) of the original p variables. These are

$$y_1 = \sum_{i=1}^{p} b_{1i}x_i, ..., y_p = \sum_{i=1}^{p} b_{pi}x_i \quad (2)$$

These can be expressed as $Y = B^T X$ where $Y = (y_1, y_2, ..., y_p)$. B^T is a $p \times p$ matrix.

3 Data Compression/Dimensionality Reduction Methods

Broadly speaking, there are two methods to perform PCA. One method includes computation of variance-covariance structure of the data and it is expressed in the form of matrix and hence known by the name matrix method. In this the matrix is diagonalized using some numerical technique such as KL Transform or Singular Value Decomposition. Second type of method works directly on the data and hence known as data method. There is no computation of covariance of the data. This method is highly suited where we have very high dimensional data and the computational cost is of prime importance. This is also suitable for real time applications. Hebbian Networks for adaptive PCA is one such example of data methods.

3.1 PCA by Eigen Decomposition

In PCA, we find the directions in the data with the most variation, i.e., the eigenvectors corresponding to the largest eigenvalues of the covariance matrix, and project the data onto these directions. The motivation for doing this is that the most of second order information is in these directions. The choice of the number of directions is often guided by principled methods. If we denote the matrix of eigenvectors sorted according to eigenvalues, by V^T, then, PCA transformation of the data is as $Y = V^T X$. The eigenvectors are called the principal components. Using the eigenvector-eigenvalue pair, the ith principal component may be written as

$$y_i = V_i^T X = v_{i1}x_1 + \cdots + v_{ip}x_p, i = 1, 2, ..., p \quad (3)$$

such that
$Var(y_i) = V_i^T C V_i = \lambda_i \qquad Cov(y_i, y_k) = 0,\ i \neq k$
We have

$$Y = V^T X = (V_1, V_2, ..., V_p)^T X \quad (4)$$

$$Var(Y) = VCV^T = diag(\lambda_1, \lambda_2, ..., \lambda_p) \quad (5)$$

We can retain the maximum information by retaining the coordinate axes that have largest eigenvalues and delete those that have less information. This technique involves

- Gather x_i where $i = 1, 2, \ldots, p$.
- Compute the mean $\bar{x}$ and subtract it to obtain $x_i - \bar{x}$.
- Compute the covariance matrix $C_{ij} = (x_i - \bar{x})(x_i - \bar{x})^T$.
- Determine the eigenvectors and eigenvalues of covariance matrix C such that $CV = \Lambda V$ where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_p)$, a diagonal matrix defined by the eigenvalues of matrix C and $V = (V_1, V_2, ..., V_p)$ be the associated eigenvectors.
- Sort the eigenvalues and corresponding eigenvectors such that $\lambda_1 \geq \lambda_2 \geq, \cdots, \geq \lambda_p$.
- Select the first $l \leq p$ eigenvectors and discard $p - l$ eigenvectors to find the data in the new directions.

If the orthogonal matrix $V = (V_1, V_2, ..., V_p)$ contains the eigenvectors of C, then C can be decomposed as $C = V \Lambda V^T$ where Λ is a diagonal matrix of eigenvalues.

3.2 Singular Value Decomposition

Singular Value Decomposition (SVD) is an important topic in linear algebra. PCA is closely related to singular value decomposition (SVD). From the numerical perspective, it is a better method and is applied directly to data matrix. Suppose, we have a matrix A with n rows and m columns, with rank r and $r \leq n \leq m$. This matrix A can be factorized into three matrices [9, 18, 19, 8](figure 1).

$$A = USV^T \quad (6)$$

where matrix U is an $n \times n$ orthogonal matrix

$$U = [u_1, u_2, u_3, \ldots, u_r, u_{r+1}, \ldots, u_n] \quad (7)$$

The column vectors of U that are u_i, for $i = 1, 2, \ldots, n$ form an orthonormal set

$$u_i^T u_j = \delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

and and matrix V is an $m \times m$ orthogonal matrix

$$V = [v_1, v_2, v_3, \ldots, v_r, v_{r+1}, \ldots, v_m] \quad (9)$$

The column vectors of V that are v_i, for $i = 1, 2, \ldots, m$ form an orthonormal set

$$v_i^T v_j = \delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

and S is an $n \times m$ diagonal matrix with singular values (SV) on the diagonal. The v_i's and u_i's are called right

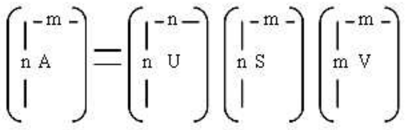

Figure 1: Factorization of A to USV^T

and left singular-vectors of A. We can define (6) as that any matrix can be converted into three, orthogonal $(n \times n)$, diagonal $(n \times m)$, and orthogonal $(m \times m)$ matrices. The numbers in figure 2, $\sigma_1, \sigma_2, \ldots, \sigma_r, \sigma_{r+1}, \ldots, \sigma_m$

$$\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & \sigma_r & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \sigma_{r+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & \sigma_m \end{bmatrix}$$

Figure 2: Singular Matrix S

are called the singular values of the data matrix A where

$$\sigma_1 \geq \cdots \geq \sigma_r > 0, \sigma_{r+1} = \cdots = \sigma_m = 0 \quad (11)$$

The number of nonzero singular values equals the rank of the matrix A. If we multiply both sides of the equation $A = USV^T$ by A^T we get
$A^T A = (USV^T)^T (USV^T) = VS^T U^T USV^T$
Since $U^T U = I$, we get

$$A^T A = VS^T SV^T = VS^2 V^T \quad (12)$$

So V diagonalizes $A^T A$, it means that columns v_i's of V (right singular vectors), are the eigenvectors of $A^T A$ $(m \times m \quad matrix)$. The eigenvalues are the square of the elements of S (Singular Values). Similarly if we multiply both sides of A on the right by A^T, we get
$AA^T = (USV^T)(USV^T)^T = USV^T V S^T U^T$
Since $V^T V = I$, we get

$$AA^T = (USS^T U^T) = US^2 U^T \qquad (13)$$

It follows that columns u_i's of U (left singular vectors), are the eigenvectors of AA^T $(n \times n \quad matrix)$. The eigenvalues are the square of the elements of S (Singular Values).

3.3 Hebbian Neural network Based PCA

There is a close correspondence between the behavior of neural networks and the statistical methods of PCA. A single linear neuron with a Hebbian type adaptation rule for its synaptic weights can evolve into a filter (called Maximum Eigenfilter) for the first principal component of the input distribution [14]. This single linear neuronal model can be expanded into a feedforward network with a single layer of linear neurons for the purpose of principal component analysis [12]. It gives us full PCA and provides us most significant principal components in an ordered fashion. This algorithm is called Generalized Hebbian Algorithm (GHA) or Sanger Rule. Consider the simple linear neuronal model shown in figure 3. The neuron receives a set of m input signals $x_1, x_2, \ldots, x_m$ through a corresponding set of m synapses with weights $w_1, w_2, \ldots, w_m$ respectively. The resulting output y is thus defined by $y = \sum_{i=1}^{m} w_i x_i$. According to Hebb postulate of learning, a synaptic weight w_i, varies with time, and grows strong when presynaptic signal x_i and postsynaptic signal coincide with each other. We may write

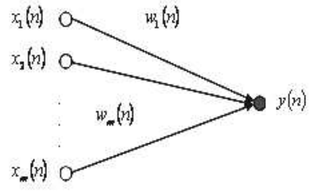

Figure 3: Signal flow graph representation of maximum eigenfilter

$$w_i(n+1) = w_i(n) + \eta y(n) x_i(n), i = 1, 2, \ldots, m \qquad (14)$$

where n denotes discrete time and η is learning-rate parameter. But, this learning rule leads to unlimited growth of the synaptic weight w_i, which is not acceptable. To overcome this problem, Oja proposed new learning rule [14]

$$w_i(n+1) = w_i(n) + \eta y(n)[x_i(n) - y(n) w_i] \qquad (15)$$

This learning rule extracts the first principal component of the input. Sanger [12] expanded this single neuronal model into a feedforward network with a single layer of linear neurons for the purpose of PCA of arbitrary size on the input. Consider the feedforward network shown in figure 4. The network has m inputs and l outputs. Each neuron in the output layer of the network is linear. The set of synaptic weights w_{ij} connecting source nodes i in the input layer to computation nodes j in the output layer, where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, l$. The output $y_j(n)$ of neuron j at time n, produced in response to the set of inputs is given by

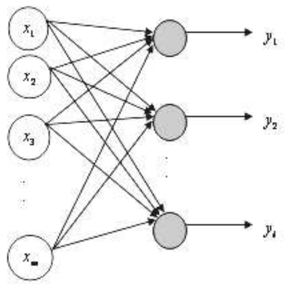

Figure 4: Feed forward network with a single layer of computation nodes

$$y_j(n) = \sum_{i=1}^{m} w_{ji}(n) x_i(n) \qquad (16)$$

The synaptic weight $w_{ji}(n)$ is adapted in accordance with a generalized form of Hebbian learning [12, 14], and is given by

$$\Delta w_{ji}(n) = \eta [y_j(n) x_i(n) - y_j(n) \sum_{k=1}^{j} w_{ki}(n) y_k(n) \qquad (17)$$

where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, l$ and $\Delta w_{ji}(n)$ is the change applied to the synaptic weight $w_{ji}(n)$ at time n and η is the learning rate parameter. GHA is an indirect method in which correlations between features of input vectors are estimated through the accumulation of repeated weight changes during training, without computing the covariance matrix of input data. It results in significant computational saving specially if the dimensionality m of the input data is very large and the required number of the eigenvectors associated

with the l largest eigenvalues of the covariance matrix is a small fraction of m. In order to perform GHA, we initialize the weights of the networks, w_{ji}, to small random values at time $n = 1$. Assign a small positive value to learning rate parameter η. Then for $n = 1$, $j = 1, 2, \ldots, l$ and $i = 1, 2, \ldots, m$, compute
$y_j(n) = \sum_{i=1}^{m} w_{ji}(n)x_i(n)$ and
$\Delta w_{ji}(n) = \eta[y_j(n)x_i(n) - y_j(n)\sum_{k=1}^{j} w_{ki}(n)y_k(n)]$
where $x_i(n)$ is the ith component of the m-by-1 input vector $x(n)$ and l is the desired number of principal components. Increment n by 1, compute $y_j(n)$ and $\Delta w_{ji}(n)$ until weights reach their steady-state values.

4 Data Compression/Dimensionality Reduction and Face Recognition using SVD, ED and Hebbian PCA

4.1 Data Compression/Dimensionality Reduction

Data compression deals with the problem of reducing the amount the data required to represent the image [3]. It is of two types, lossless and lossy data compression. Lossless data compression algorithms allow the exact original data to be reproduced from the compressed data whereas with lossy data compression algorithms the exact reconstruction of original data is not obtained. Run Length Encoding, Huffman Coding, Shannon-Fano Coding are some examples of lossless data compression techniques. Discrete Cosine Transform, Wavelet Compression, Vector Quantization are among the lossy compression techniques. The technique presented in this paper belongs to lossy data compression algorithms category. The advantage of lossy methods over lossless methods is that in some cases a lossy method can produce a much smaller compressed file than any other lossless method, while still satisfying the requirements of the application.

In the previous section we saw that the number of nonzero singular values in Singular Value Decomposition (SVD) is equal to the rank of the matrix A. In many applications, these singular values decrease quickly as the rank of the matrix is increased. We can use the rank of the matrix to remove the redundant information.

$$A = USV^T = \sum_{i=1}^{m} \sigma_i u_i v_i^T \tag{18}$$

This can be expanded and written as
$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T + 0u_{r+1}v_{r+1}^T +$
Dropping the terms where the singular values are zeros, we get

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T \tag{19}$$

Further approximation can be done by dropping more singular terms. We know that the singular values are arranged in descending order. The last terms will have least effect on the overall image. We can have first d terms and drop (r-d) sum terms to get A_d. The storage will then be

$$STR = d \times (n + m + 1) \tag{20}$$

The compression ratio for the image is

$$CR = n \times m/STR = (n \times m)/d \times (n + m + 1) \tag{21}$$

Further the quality of the compressed image is computed in terms of the Mean Square Error

$$MSE = 1/nm \sum_{y=1}^{n} \sum_{x=1}^{m} (f_A(x, y) - f_{A_d}(x, y)) \tag{22}$$

The dimensionality reduction in Eigen Decomposition PCA and Hebbian Neural Network based PCA, is obtained by reducing the number of features needed for effective data representation. We know that $Y = V^T X$, premultiply both sides of $Y = V^T X$ by the matrix V and then using the fact that $V^T = V^{-1}$, we may reconstruct the original data which is as follows

$$X = VY = \sum_{i=1}^{p} y_i v_i \tag{23}$$

The number of features is reduced by discarding those linear combinations of (23) that have small eigenvalues and retain those terms that have large eigenvalues. Let $\lambda_1, \lambda_2, \ldots, \lambda_l$ denote the largest l eigenvalues of the covariance matrix C. The approximation of original data is done by retaining the first l terms and truncating those after l terms.

$$\hat{X} = \sum_{i=1}^{l} y_i v_i \tag{24}$$

and the Mean Square Error (MSE) between the actual data X and the approximation $\hat{X}$ is calculated using (22). Face images have been considered for data compression and dimensionality reduction.

For SVD, when applied to face image for data compression, get the matrices U, S and V. Take suitable value d for the compression. Calculate the compression ratio and the quality of the image (MSE).

For ED, gather a set of face images x_i where $i = 1, 2, \ldots, p$ in the form of matrix where each column represents one image we get after vertical concatenation. Compute the mean $\bar{x}$ and subtract it to obtain $x_i - \bar{x}$. Compute the covariance matrix $C_{ij} = (x_i -$

$\bar{x})(x_i - \bar{x})^T$. Determine the eigenvectors and eigenvalues of covariance matrix C such that $CV = \Lambda V$ where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_p)$, a diagonal matrix defined by the eigenvalues of matrix C and $V = (V_1, V_2, ..., V_p)$ be the associated eigenvectors. Sort the eigenvalues and corresponding eigenvectors such that $\lambda_1 \geq \lambda_2 \geq, \cdots, \geq \lambda_p$. Select the first $l \leq p$ eigenvectors and discard $p - l$ eigenvectors to find the data in the new directions.

In Hebbian Neural Network based PCA, after gathering a set of face images x_i and computing the mean $\bar{x}$ to obtain $x_i - \bar{x}$, we initialize the weights of the network, w_{ij}, to small random values. The size of the weight matrix depends on the total number of face images and the number of eigenvectors to be retained for reconstruction. The Generalized Hebbian Algorithm of (17) will give us a matrix whose columns are the first d eigenvectors ordered by the decreasing eigenvalues.

4.2 Face Recognition

Face Recognition[7,10] has always been a hot research area. It has drawn the attention of many researchers because of its practical applications. Security systems, criminal identification, human-computer interaction, entertainment, and video surveillance (etc.) are among the most common applications that fascinated many researchers. For face recognition, there are broadly two categories namely i) Holistic or Appearance Based techniques that use the whole face as the input to recognition system, and ii) Feature Based techniques where the local features such as the eyes, and local statistics are fed to the recognition system. Principal Component Analysis has been successfully used for face recognition.

Consider a set of N sample images $\Gamma_1, \Gamma_2, \ldots, \Gamma_N$ taking values in an n-dimensional image space. Assume each face image has $m \times n = M$ pixels, and is represented as a $M \times 1$ column vector. A training set Γ with N number of face images of known individuals forms an $M \times N$ matrix.

$$\Gamma = [\Gamma_1, \Gamma_2, \ldots, \Gamma_N] \tag{25}$$

ψ is the mean image of all the samples.

$$\psi = 1/N \sum_{i=1}^{N} \Gamma_i \tag{26}$$

subtract ψ from all sample images to get a new set $F = [\phi_1, \phi_2, \ldots, \phi_N]$ where $\phi_i = \Gamma_i - \psi$. Principal Component Analysis is applied to the new set F. We get a set of N orthonormal vectors V_i. The kth vector V_k is chosen such that

$$\lambda_k = 1/N \sum_{i=1}^{N} (V_k^T \phi_i)^2 \tag{27}$$

is maximum subject to

$$V_j^T V_k = \begin{cases} 1, & \text{if } j = k \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

The vectors V_k and the scalars λ_k are the eigenvectors and the eigenvalues, respectively, of the covariance matrix

$$C = 1/N \sum_{k=1}^{N} \phi_i \phi_i^T = FF^T \tag{29}$$

Only d number of eigenvectors $V = [V_1, V_2, ..., V_d]$ of C is chosen that correspond to the d largest eigenvalues $[\lambda_1, \lambda_2, \ldots, \lambda_d]$

For SVD, we compute singular value decomposition of F. This process gives us three matrices U, S and V. Choose first d columns of the matrix U that are the eigenvectors of matrix FF^T.

For Hebbian Neural Network PCA, initialize the weights of the network, w_{ij}, to small random values. The size of the weight matrix W depends on the total number of face images and the number of eigenvectors (say d) to be retained. The Generalized Hebbian Algorithm of (17) will give us a matrix whose columns are the first d eigenvectors ordered by the decreasing eigenvalues.

Thus for performing the dimensionality reduction on input data, we need to compute the eigenvalues and eigenvectors and then project the data orthogonally onto the subspace spanned by the eigenvectors belonging to the dominant eigenvalues and leaving those that possess less information.

For a given set of eigenvectors, $[V_1, V_2, \ldots, V_d]$ in ED, $[U_1, U_2, ..., U_d]$ of U in SVD and the d columns of the matrix $(F \times W^T)$, in Hebbian Neural Network PCA, project the new image f_{new} onto the eigenvectors to give [7]

$$w_k = U_k^T f_{new} \quad for \ \ k = 1, 2, \ldots, d \tag{30}$$

The w_k form a weight distribution vector

$$w = [w_1, w_2, \ldots, w_d] \tag{31}$$

Euclidean norm has been used as the measure of similarity. The test image (new image) weight vector is matched with those of known (training) images. A match is said to occur if the image, out of known images, having the minimum Euclidean distance among all known images, is of the same class to which the test image belongs to. The flowchart for face recognition is shown in figure 5.

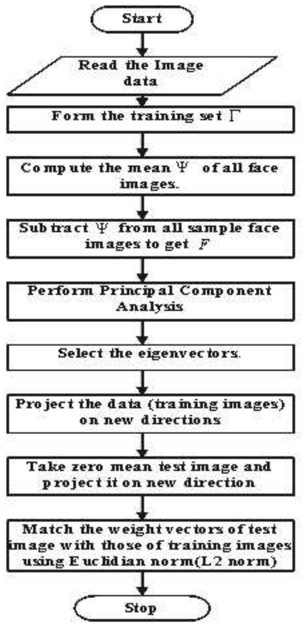
Start
Read the Image data
Form the training set Γ
Compute the mean Ψ of all face images.
Subtract Ψ from all sample face images to get F
Perform Principal Component Analysis
Select the eigenvectors.
Project the data (training images) on new directions
Take zero mean test image and project it on new direction
Match the weight vectors of test image with those of training images using Euclidian norm(L2 norm)
Stop

Figure 5: Flow chart of face recognition

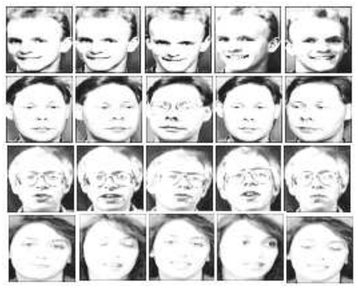

Figure 6: Some images of ORL database

Figure 7: Original image

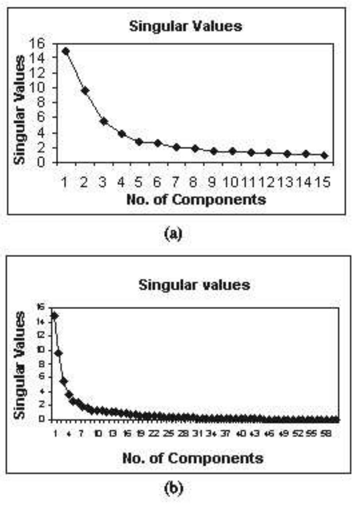

Figure 8: Singular values (a) first 15 components (b) all components

5 Experimental Results and Discussion

We use ORL face database for experimentation in this paper [11]. It is composed of 400 images with each image having a resolution 92x112. As many as 40 different persons (subjects/classes) are contained in the database and each person has his/her 10 different images. Figure 6 shows some sample images of this database. These images vary in terms of facial expressions and facial details. These images have been taken at different times, varying lighting slightly, facial expressions (open/close eyes, smiling/no-smiling), and facial details (glasses/ no-glasses). All the images are taken against a dark homogeneous background and the subjects are in up-right, front position with slight left right rotation. For computational simplicity, the original image 92x112 was resized to 60x60 prior to further processing of the face image.

5.1 Data Compression/Dimensionality Reduction

The first experiment was performed to show the results of image reconstruction for different singular values. Figure 7 shows the original image. Figure 8 shows the graph between singular values and the number of components. It is not easy to determine the number of components d to be used. We can use energy fraction to select the number of components. The other way to determine the number of components could be the characteristics of the singular values. When the singular values stabilize, the remaining components are usually are of no use as they are contaminated by the noise. In figure 8, we see that from the component number 9 and above, the singular values are almost constant, indicating that d should range 8 to 10.

The image was reconstructed using different number (d) of singular values. The reconstructed images are shown in figure 9. Figure 9 clearly shows that for less

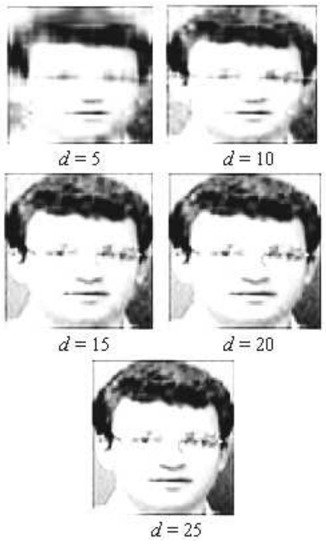

Figure 9: Reconstructed image for different d values

value of d, the images are blurry.The quality increases and the images are close to the original as the d value increases. Table 1 shows the storage space and the error measures for the face image. Table 1 shows that as d is

Table 1: Results for Image Compression

d	MSE	$STR(Bytes)$	CR
Original	-	3600	-
05	1.952900	0605	5.9507
10	0.780850	1210	2.9752
15	0.346040	1815	1.9835
20	0.158330	2420	1.7876
25	0.077049	3025	1.1901

increased, it results in lesser value of MSE and hence we get better quality image. The reconstructed image is more close to the original one but at the same time the storage space is increased. In order to achieve better compression ratio, the value of d must be low. From table 1 we see that the compression ratio is 1.1901 when d =25, the reconstructed image is closer to original image as the error is 0.077049.

In the next experiment, 5 out of total of 10 images were taken from each class for a total of 20 classes. PCA was applied on the set of 100 images (matrix form, one column representing one image). Figure 10 shows a graph between the eigenvalues and the number of eigenvectors. Again, in order to select the number of eigen-

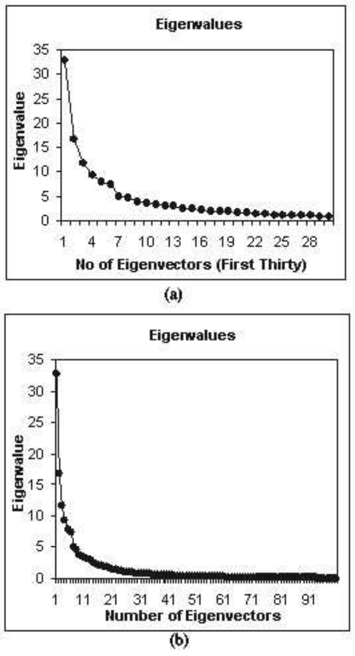

Figure 10: Eigenvalues (a) first 30 components (b) all components

vectors, we need to calculate either the total variance we want to retain or we need to refer to the graph shown in figure 10. The graph shows the plot of jth eigenvalue corresponding to the jth eigenvector. The number of eigenvectors d is chosen to be the point at which the line on the graph is steep to the left but not steep to the right. In the figure 10 we see that such point could be the component number 12. The eigenvalues are almost constant beyond 12, indicating that d should be 12. Table 2 shows the Total Variance Contribution Rate corresponding to the number of eigenvalues. It is calculated as

$$TVCR(\lambda_1, \lambda_2, \ldots, \lambda_d) = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \qquad (32)$$

where p is the total number of eigenvalues and d is the number of selected eigenvalues. From Table 2, it is clear that if we consider number of eigenvalues as 10, then TVCR value is 63.8 percent and if the number of eigenvalues is 80 then TVCR value is 98.6 percent. Depending upon the value of the TVCR to be retained, we can select the number of eigenvectors.

Table 2: TVCR for Different Number of Eigenvalues

No. of Eigvalues	90	80	60	40	20	10
TVCR	99.5	98.6	95.6	90.3	78.6	63.8

The experiment was performed for reconstruction of the images. Mean Square Error (MSE) value was determined for three different sets (1 image, 3 images, 5 images/class) for a total of 20 classes. Figure 11 shows the reconstructed image for a set of 5 training images for first 20 classes. Upon applying PCA, we get a total of as many as 100 eigenvectors for the set of images under consideration. The images have been reconstructed for different number of eigenvectors. MSE value for different sets of training images for a total of 20 classes is as shown in figure 12. The results show that Mean Square Error between the original and reconstructed image decreases as the number of eigenvectors increase. The reconstructed image approaches more towards the original one. Further if there is more than one image per class in the training set of images, it results in reduction of error as compared to that we have when single image per class is used.

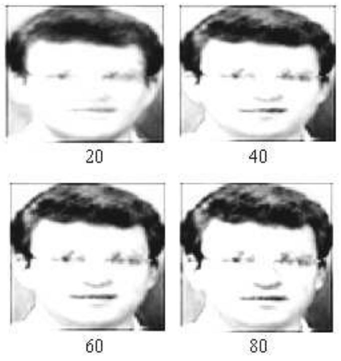

Figure 11: Reconstructed image for different number of eigenvectors

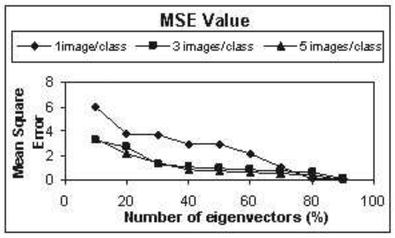

Figure 12: MSE for different number of eigenvectors

The next experiment was performed for Hebbian Neural Network PCA. Here also MSE between the original and reconstructed image was determined for the above said three sets of training images. Prior to finding the MSE value, we took a set of 10 classes, 5 images per class and it was trained. A particular image was reconstructed using 50 percent of the total number of eigenvectors and MSE between the original and the reconstructed image was found for different values of the learning rate parameter. This experiment was performed to find the value of the learning rate for which the error was minimum. The number of iterations was taken as 100. From figure 13, it is clear that the MSE value is minimum for $\eta = 0.0010$. A similar experiment was performed to see the effect of number of iterations on the MSE value for learning rate 0.0010. The results are shown in figure 14. Figure 14 shows that as the number of iterations is increased beyond 100, it results in very little change in the MSE value. So the number of iterations was kept as 200 for further experimentation.

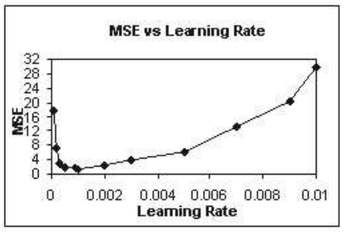

Figure 13: MSE for different values of η

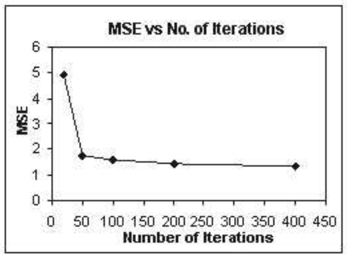

Figure 14: MSE for different values of iterations

This experimentation included the reconstruction of the image. The neural network considered a set of 5 training images for first 20 classes for a total of 100 neurons at the input layers. The images were reconstructed considering 20, 40, 60, and 80 neurons at the output layer and hence the number of eigenvectors respectively. Figure 15 shows the reconstructed images.

Hebbian Neural Network was also trained for three dif-

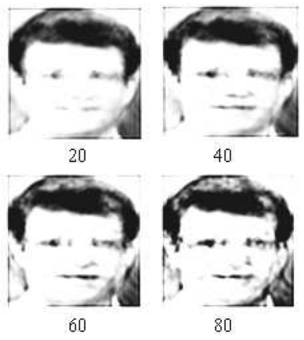

Figure 15: Reconstructed image for different numbers of eigenvectors

ferent sets (1 image, 3 images, 5 images/class) for a total of 20 classes with the number of neurons at the output layer as 10, 20, 30, 40, 50, 60, 70, 80 and 90 percent of the total number of eigenvectors respectively. Figure 16 shows the MSE value for the experiment. The re-

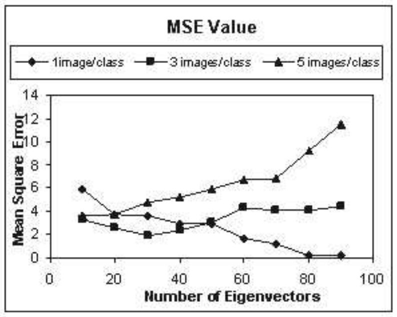

Figure 16: MSE for different number of eigenvectors

sults show that if only one image per class is used for training the network, the reconstruction of the image results in lesser error as the number of eigenvectors increase. But this is not true when the number of images per class is increased for training. It increases the MSE value as the number of eigenvectors increase. Whereas keeping one image per class and increasing the number of classes does indicate that as the number of eigenvectors increase, the error is reduced as shown in figure 17.

5.2 Face Recognition

A set of five images from each class for total of 10, 20 and 40 classes was used as the training images and a set of remaining non overlapping five images was used for testing. Figure 18 shows the set of training images and

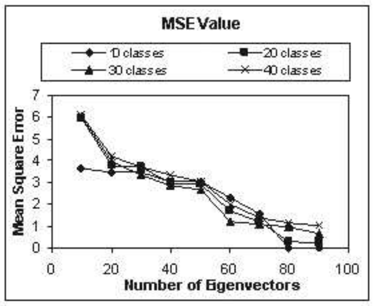

Figure 17: MSE for different number of eigenvectors

test images. The recognition rate was measured for 20,

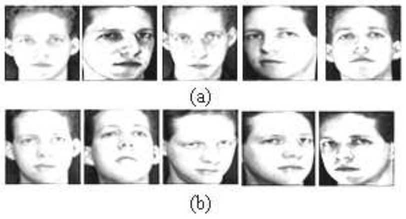

Figure 18: (a) Training images (b) Test images

40, 60 and 80 percent of the total number of eigenvectors. The number of classes was varied from 10 to 20 to 40. We get same results with PCA-ED or PCA-SVD method. But the results differ for PCA- Hebbian Neural Network. These are as shown in figures 19 and 20 respectively.

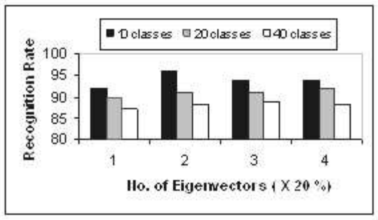

Figure 19: Recognition rate for different number of eigenvectors for PCA-ED/PCA-SVD

As we see (figures 19 and 20) that the recognition rate decreases as the number of classes are increased. This may be due to the fact that as the number of images (classes) increase, there are more chances of image being wrongly recognized because of more similar faces. It has also been observed that for 10 classes, the recognition rate is almost same be it PCA using ED/SVD or PCA for Hebbian Neural Network method. Moreover

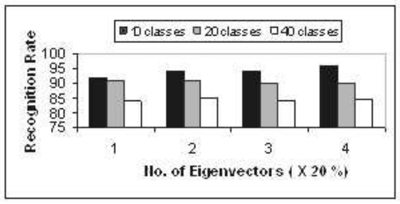

Figure 20: Recognition rate for different number of eigenvectors for PCA-Hebbian NN

for 40 classes, the recognition rate remains almost same for different number of eigenvectors though it is less in PCA for Hebbian Neural Network method than that of PCA using ED/SVD.

6 Conclusions

This paper presents a review of various methods, Eigen Decomposition, Singular Value Decomposition and Hebbian Neural Network to perform Principal Component Analysis. These methods have been applied to data compression/dimensionality reduction and face recognition. Eigen Decomposition and Singular Value Decomposition are related techniques. SVD has the numerical advantage over ED. For a prescribed accuracy of computation, SVD procedure requires half the numerical precision of the ED procedure. ED and SVD are known as batch methods of computations. The Hebbian-based Neural Network is of adaptive method of computation. The results shows that as the number of singular values are increased, it results in lesser value of MSE and hence we get better quality image. The reconstructed image is more close to the original one but at the same time the storage space is increased. The results (table 1) reveal that the compression ratio is 1.1901 when d = 25, the reconstructed image is closer to original image as the error is 0.077049. It gives a very good compression ratio (5.9507) when d = 5 only but the error (1.952900) is more. The results also show that the two methods, ED and Hebbian Neural Network yield almost same output in terms of MSE if only one image per class is considered. But as the number of images per class increase, ED performs better than Hebbian Neural Network method. So far as recognition rate is concerned, we get the same results irrespective of which of the two methods (ED or SVD) is used. It has also been observed that for 10 classes, the recognition rate is almost same be it PCA using ED/SVD or PCA for Hebbian Neural Network method. Moreover for 40 classes, the recognition rate remains almost same for different number of eigenvectors though it is less in PCA for Hebbian Neural Network method than that of PCA using ED/SVD.

7 References

[1] Anderson, An introduction to multivariate statistical analysis, Wiley, 2nd edition, New York, 1958

[2] Li, S.Z, Application of the correlation analysis in feature selection, Journal of Test and Measurement Technology, No. 1, 2001.

[3] Mrak M, Grgic S, and Grgic M., Picture Quality measure in image compression systems, IEEE Region 8 EUROCON, Vol. 1, p. 233-236, 2003.

[4] Lewis, M. S., Factor analysis and related techniques, SAGE Publications, Vol.5, p.157-194, 1994.

[5] Haykin, S., Neural networks-A comprehensive foundation, Pearson Education, Vol.2, 2004.

[6] Leon, S., Linear algebra with applications, McMillian Publishing Company, New York, 1996.

[7] Turk, M. A., Pentland P. A., Face recognition using eigenface method, IEEE Conference on Computer Vision and Pattern Recognition, p.586-591, 1991.

[8] Cao, L., Singular value decomposition applied to digital image processing, Division of Computing Studies, Arizona State University Polytechnic Campus, Arizona, May 2007.

[9] Golub, G.H., Loan, C. F. V., Matrix computations, Johns Hopkins University Press, Baltimore, USA,1983.

[10] Jun Zhang, Yong Yan, Lades, M., Faces recognition: eigenface, elastic matching and neural net, Proceedings of the IEEE, Vol. 85, No. 9, p.1423-1435, 1997.

[11] http://www.cl.cam.ac.uk/ ORL face database.

[12] Sanger, T.D., Optimal unsupervised learning in single layer linear feedforward neural networks, Journal of Neural Networks, Vol. 2, p.459-473, 1989.

[13] Delichere, M., Daniel, Neural dimensionality reduction for document processing, Proceedings of ESANN, p.211-216, 2002.

[14] Oja, E, A simplified neuron model as a principal

component analyzer, Journal of Mathematics and Biology, Vol. 15, p. 267-273, 1982.

[15] Jolliffe, I.T., Principal component analysis, Springer Verlag, 1986.

[16] Sirovich, L., Kirby, M., Low-dimensional procedure for characterization of human faces, Journal of Optical Society, America, Vol. 4, p. 519-524, 1987.

[17] Hotelling, H., Analysis of a complex of statistical variables into principal components, J. Educ. Psychol, Vol. 24, p. 417-441, 498-520, 1993.

[18] Stewart, G. W., On the early history of singular value decomposition, SIAM Review, Vol. 35, No. 4, p. 551-566, 1993.

[19] Wall, M.E., Andreas Rechtsteiner, Luis M Rocha, Singular value decomposition and principal component analysis, In: A practical approach to microarray data analysis. Berrar, D.P., Dubitzky, W., Granzow, M., (Eds.) p. 91-109, Kluwar, Norwell, MA, 2003.