DIGITAL IMAGE PROCESSING TECHNIQUES FOR FACE RECOGNITION

FERNANDO ROMAN

ABSTRACT. Face recognition is one of many applications of digital image processing. It is concerned with the automatic identification of an individual in a digital image. There are many algorithms through which this process can be carried out. One of these algorithms compresses a database of face images and keeps only the data useful for facial approximation. Any face image can then be approximated using only the information contained in the compressed database, even if the image was not in the original database. In this project we study this algorithm, its underlying mathematical frameworks (SVD and PCA), its interpretation in terms of face images and, its implementation through the use of MATLAB.

1. Introduction

A gray scale digital image of size $M \times N$, $M, N \in \mathbb{N}$, can be mathematically defined as a matrix with entries $f(x, y)$, $x = 0, 1, ..., M - 1$, $y = 0, 1, ..., N - 1$, where the value $f(x, y)$ represents the intensity or gray level of the image at the pixel $(x, y)$. The intensity values of a gray-scale image range in a discrete interval between two numbers $a$ and $b$ ($a < b$) where $a$ represents the lowest intensity (i.e. black), $b$ represents the highest intensity (i.e. white) and all the values in between represent different levels of gray from black to white. The numbers $a$ and $b$ depend on the class of the image; a very commonly used range is the set of intensities $\{0, 1, ..., 255\}$ where 0 represents black and 255 represents white.

Using this mathematical interpretation of an image as a matrix, we are able to process images to improve its pictorial information and compress images for data storage and autonomous machine interpretation. Manipulations of an image to improve pictorial information include intensity transformations, spatial filtering, frequency filtering, among others. Manipulations for storage and autonomous machine interpretation include other machinery like data compression. The mathematical background of some of these manipulations are discussed in Section 2.

The main goal of this article is to discuss digital image processing techniques for face recognition. We will see that many of the mathematical tools needed for the face recognition process come from linear algebra; in particular singular value

Date: July 28, 2013.
2010 Mathematics Subject Classification. Primary 15A18; Secondary 62H25.

Key words and phrases. Digital image, matrix, singular value decomposition, principal component analysis, face recognition.

Supported by the Center for the Integration of Undergraduate, Graduate, and Postdoctoral Research. Many thanks to Professor Virginia Naibo Ph.D, and graduate student Vincent Pigno for their guidance in this work, and to graduate student Jodi Herbert for her help with LaTex.
decomposition (SVD) and principal component analysis (PCA) will be the most commonly used tools in this article.

In section 3 we present a detailed mathematical description of SVD and PCA. Computer implementation for face recognition that uses SVD and PCA are shown in Sections 4 and 5.

2. IMAGE MANIPULATIONS

Let \( f(x, y), x = 0, 1, ..., M - 1, y = 0, 1, ..., N - 1 \), represent an \( M \times N \) gray scale digital image. For the purposes of this discussion we assume that the intensity values range between \( a = 0 \) and \( b = 1 \).

2.1. Spatial Domain Techniques. Manipulations in the spatial domain are manipulations applied directly to the intensity values of the image (i.e. \( f(x, y) \)). Examples include intensity transformations and spatial filtering, among others.

a. Intensity transformations: Let \( T : [0, 1] \rightarrow [0, 1] \), consider the image represented by the \( M \times N \) matrix \( g(x, y) = T(f(x, y)), x = 0, ..., N - 1, y = 0, ..., M - 1 \). The function \( T \) is called intensity transformation and, depending on its choice, the image \( g \) may result in an enhanced version of \( f \) for better human interpretation. Some examples of basic intensity transformations are:

i. Negative of an image: inverts the intensity values of the pixels (i.e. sends black to white, white to black and “inverts” the gray scale values). \( T \) can be defined as:
\[
T(u) = 1 - u.
\]

ii. Gamma transformation: Either maps a narrow range of dark input values into a wider range of output values, or maps a wide range of input values into a narrower range of output values. \( T \) can be defined as:
\[
T(u) = c(u + \alpha)^\gamma,
\]
where \( c, \alpha \) and \( \gamma \) are appropriate parameters.

iii. Intensity Level Slicing: Highlight a specific range of intensities in an image. It can be done highlighting the desired range and leaving the rest of the image the same, or highlighting the desired range and changing everything else to a specific intensity. \( T \) can be defined by:
\[
T(u) = \begin{cases} 
I_1, & p \leq u \leq q, \\
I_2, & \text{otherwise,}
\end{cases}
\]
where \([p, q]\) is the desired range to highlight and, \( I_1 \) and \( I_2 \) are appropriately chosen intensity values.
Figure 1. The original image (on the left) is dark, due to background light. To the right, we can see the image after applying a gamma transformation with $\gamma = 0.5$.

b. Spatial Filtering: Spatial filtering can be applied either to sharpen an image and increase detail or to smooth an image and reduce noise. There are two main types of spatial filtering: linear and nonlinear. Linear spatial filtering is carried out by applying a $p \times q$ mask $W$ to a neighborhood of each pixel in the image. These masks are often called filters and are applied through correlation or convolution.

An example of a linear filter is the averaging filter which changes the intensity values $f(x, y)$ for the average intensity values of pixels in a neighborhood of $(x, y)$. More precisely, the new intensity value at $(x, y)$ is given by:

$$\frac{1}{(2a+1)(2b+1)} \sum_{s=-a}^{a} \sum_{t=-b}^{b} f(x+s, y+t)$$

where $a$ and $b$ are fixed non-negative integers. This filter corresponds to a mask $W$ of size $(2a+1) \times (2b+1)$ and with all entries equal to $\frac{1}{(2a+1)(2b+1)}$. This process smooths the image and reduces noise.

An example of a non-linear filter is the first derivative of the image, used to sharpen the image and increase detail. The two first partial derivatives of an image can be defined as:

$$\frac{\partial f}{\partial x}(x, y) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \otimes f(x,y)$$

and

$$\frac{\partial f}{\partial y}(x, y) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \otimes f(x,y),$$
where $\otimes$ is the correlation defined by

$$(W \otimes f)(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} W(i, j)f(x + i, y + j)$$

for $p \times q$ matrix $W$ with $p = 2a + 1$ and $q = 2b + 1$.

Then, the first derivative of $f(x, y)$ is given by the image

$$f'(x, y) = \sqrt{\left(\frac{\partial f}{\partial x}(x, y)\right)^2 + \left(\frac{\partial f}{\partial y}(x, y)\right)^2}$$

for $x = 0, 1, ..., M-1$, $y = 0, 1, ..., N-1$.

![Original Image](image1.png) ![Filtered Image](image2.png)

**Figure 2.** The image to the left has some salt and pepper noise. To the right, the image after filtering with an averaging mask.

2.2. **Frequency Domain Techniques.** In this section we give an overview of the concepts of Fourier transform and discrete Fourier transform. We also mention some applications of the latter in digital image processing.

(1) Fourier transform of functions of a continuous variable

(a) Fourier representation of periodic functions

Let $f(x)$ be a periodic function with period $p$ and integrable in the interval $[-p, p]$. Then the Fourier series of $f$ is defined by

$$\sum_{n=-\infty}^{\infty} C_n e^{\frac{2\pi in}{p}x}$$
where
\[ C_n = \frac{1}{p} \int_{-p/2}^{p/2} f(x) e^{-2\pi in x/p} \, dx. \]

Under certain assumptions on \( f \), it follows that \( f(x) = \sum_{n=-\infty}^{\infty} C_n e^{2\pi in x/p} \) for almost every \( x \in \mathbb{R} \).

(b) Fourier representation of non-periodic functions

If the function \( f(x) \) is integrable in \( \mathbb{R} \), then its Fourier transform is defined by
\[ \hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x} \, dx, \quad w \in \mathbb{R} \]

Under certain assumptions on \( f \) it follows that
\[ f(x) = \int_{-\infty}^{\infty} \hat{f}(\omega) e^{2\pi i \omega x} \, d\omega \]
for almost every \( x \in \mathbb{R} \).

The definitions presented above can be easily generalized to functions of several variables.

(2) Discrete Fourier Transform (DFT)

The discrete Fourier transform is defined for periodic functions of discrete variables. A digital image of size \( M \times N \) can be thought of as a function \( f(x, y) \), \( x, y \in \mathbb{Z} \), that is periodic in \( x \) of period \( M \) and periodic in \( y \) of period \( N \). Since our goal in this section is to show some applications of the DFT to digital images, we only give the definition of the 2-dimensional DFT. Analogous definitions apply to functions of different number of variables.

Let \( f(x, y) \) be a function defined in \( \mathbb{Z}^2 \) that is periodic in \( x \) of period \( M \) and periodic in \( y \) of period \( N \). This is,
\[ f(x + M, y + N) = f(x, y), \quad \forall x, y \in \mathbb{Z}. \]
The discrete Fourier transform of \( f \) is defined by
\[ \hat{f}(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-2\pi i \left( \frac{xu}{M} + \frac{yu}{N} \right)}, \quad u, v \in \mathbb{Z}, \]
and the inverse discrete Fourier transform of \( f \), is defined by
\[ \tilde{f}(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \hat{f}(u, v) e^{2\pi i \left( \frac{xu}{M} + \frac{yu}{N} \right)}, \quad x, y \in \mathbb{Z}. \]
As expected, the DFT and the inverse DFT satisfy $\hat{f} = \hat{\hat{f}} = f$. Then, $f$ can be recovered from $\hat{f}$ through the formula

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \hat{f}(u, v)e^{2\pi i (\frac{ux}{M} + \frac{vy}{N})}, \quad x, y \in \mathbb{Z}.$$ 

This means that $f$ is a superposition of the wave functions

$$C_{uv}(x, y) = e^{2\pi i (\frac{ux}{M} + \frac{vy}{N})}, \quad x, y \in \mathbb{Z},$$

with the coefficients $\hat{f}(u, v)$ corresponding to low frequency components if $u$ and $v$ are small, and high frequency components if $u$ and $v$ are close to $\pm \frac{M}{2}$ and $\pm \frac{N}{2}$ respectively. Note that $\hat{f}(u, v) = |\hat{f}(u, v)|e^{i\phi(u, v)}$ where $|\hat{f}(u, v)|$ represents a magnitude and $\phi(u, v)$ represents an angle; $\phi(u, v)$ is called the phase angle of $\hat{f}$ and $|\hat{f}|$ is the spectrum of $\hat{f}$. These two, the phase angle and the spectrum, are useful in applications like frequency filtering and image restoration.

- **Lowpass filters**
  
  As the name suggests this kind of filter does not affect the “low frequency components” and it passes the “high frequency components” with some attenuation determined by the filter. The frequency that divides the high and low frequencies is said to be the “cut-off” frequency. This cut-off frequency can be arbitrarily determined depending on the desired result. By attenuating the high frequencies, lowpass filters can smooth an image and reduce noise. An example of a simple lowpass filter is the $P \times Q$ ideal filter which is defined by

$$H(u, v) = \begin{cases} 1, & D(u, v) \leq D_0 \\ 0, & D(u, v) > D_0 \end{cases}$$

where $D_0$ is the cut-off frequency and $D(u, v) = [(u - \frac{P}{2})^2 + (v - \frac{Q}{2})^2]^{\frac{1}{2}}$ is the distance from $(u, v)$ to the center of the DFT of the image after it has been centered.

- **Highpass filters**
  
  Analogously, this kind of filter leaves the high frequency components unaffected and passes the low frequency components with some attenuation. This process may be used to sharpen the image improving detail. An example of high-pass filter is the $P \times Q$ Gaussian filter defined by

$$H(u, v) = 1 - e^{-\frac{[D(u, v)]^2}{2D_0^2}},$$

where, again, $D_0$ is the cut-off frequency and $D(u, v)$ is the distance from $(u, v)$ to the center of the DFT of the image.
3. SINGULAR VALUE DECOMPOSITION AND PRINCIPAL COMPONENT ANALYSIS

In this section we give the mathematical background needed for face recognition techniques described in Section 4.

3.1. SINGULAR VALUE DECOMPOSITION.

A. Singular Values of a Matrix

Given an $M \times N$ matrix, the matrix $A^T A$ is an $N \times N$ symmetric matrix and, hence, it can be orthonormally diagonalized. Moreover, $A^T A$ is positive semi-definite; that is: $\forall \mathbf{x} \in \mathbb{R}^N$, $\mathbf{x}^T (A^T A) \mathbf{x} \geq 0$. Indeed,

$$\mathbf{x}^T (A^T A) \mathbf{x} = (A \mathbf{x})^T A \mathbf{x} = \|A \mathbf{x}\|^2 \geq 0,$$

where $\|\mathbf{u}\| = \sqrt{u_1^2 + \ldots + u_N^2}$ for $\mathbf{u} = (u_1, \ldots, u_N) \in \mathbb{R}^N$. In particular, this implies that all the eigenvalues of $A^T A$ are nonnegative.

Denote by $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N \geq 0$ the eigenvalues of $A^T A$ arranged in decreasing order, and let $v_1, v_2, \ldots, v_N$ be an orthonormal basis for $\mathbb{R}^N$ consisting of corresponding eigenvectors of $A^T A$. Then for $1 \leq i \leq N$ we have

$$\|A v_i\| = \sqrt{\lambda_i}.

The number $\sigma_i = \sqrt{\lambda_i}$ is called the $i^{th}$ singular value of $A$.

B. Decomposition

We will show that an $M \times N$ matrix $A$ can be factored as $A = U \Sigma V^T$, where $U$ is an $M \times M$ orthogonal matrix, $V$ is an $N \times N$ orthogonal matrix, and $\Sigma$ is an $M \times N$ matrix such that $\Sigma_{ij} = 0$ if $i \neq j$ and $\Sigma_{ii} = \sigma_i$, $i = 1, \ldots, \min\{M, N\}$, being $\sigma_i$ the $i^{th}$ singular value of $A$. 

**Figure 3.** To the left, the original image. In the Center, the discrete Fourier transform of the image after being centered, and to the right the image after applying an ideal high-pass filter.
**Lemma 1. [see for instance [2]]**

Let $A$ be an $M \times N$ matrix. Suppose $v_1, v_2, ..., v_N$ is an orthonormal basis of $\mathbb{R}^N$ consisting of eigenvectors of $A^T A$ arranged so that the corresponding eigenvalues satisfy $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N$; and suppose $A$ has $p$ nonzero singular values. Then, $Av_1, Av_2, ..., Av_p$ is an orthogonal basis for $\text{Col}(A)$ and $\text{Rank}(A) = p$

**Proof.** For $i \neq j$, $v_j$ and $\lambda_i v_i$ are orthogonal, so we get:

$$(Av_j)^T (Av_i) = v_j^T A^T Av_i = v_j^T (\lambda_i v_i) = 0$$

Hence $Av_1, Av_2, ..., Av_N$ is an orthogonal set. Also since there are $p$ nonzero singular values and the length of $Av_i$ is the $i^{th}$ singular value, $Av_i = 0$ iff $p < i \leq N$. So $Av_1, Av_2, ..., Av_p$ is linearly independent and it is in $\text{Col}(A)$. Now let $y \in \text{Col}(A)$, then $y = A x$ where $x = c_1 v_1 + c_2 v_2 + ... + c_N v_N$, which gives $y = A x = c_1 Av_1 + c_2 Av_2 + ... + c_N Av_N = c_1 Av_1 + c_2 Av_2 + ... + c_p Av_p$ thus $y \in \text{span}\{Av_1, Av_2, ..., Av_p\}$ which shows that $\{Av_1, Av_2, ..., Av_p\}$ is a basis for $\text{Col}(A)$ and therefore $\text{Rank}(A) = \dim(\text{Col}(A)) = p$.

**Theorem 2. [see for instance [2]]**

If $A$ is an $M \times N$ matrix, then there exist a $M \times M$ orthogonal matrix $U$ and a $N \times N$ orthogonal matrix $V$ such that $A = U \Sigma V^T$, where $\Sigma$ is the $M \times N$ diagonal matrix whose entries in the main diagonal are the singular values of $A$ arranged in decreasing order.

**Proof.** Let $p = \text{Rank}(A)$ and for $i = 1, ..., N$, let $\lambda_i$ and $v_i$ be as in Lemma 1. For $1 \leq i \leq p$, let $u_i = \frac{Av_i}{\|Av_i\|} = \frac{1}{\sigma_i} Av_i$, then $\{u_1, ..., u_p\}$ is an orthonormal basis of $\text{Col}(A)$, and can be extended to an orthonormal basis $\{u_1, ..., u_M\}$ of $\mathbb{R}^M$. We define $U = [u_1 u_2 ... u_M]$ and $V = [v_1 v_2 ... v_N]$ where $u_i$ and $v_i$ are the $i^{th}$ column of $U$ and $V$ respectively.

Clearly $U$ and $V$ are orthogonal matrices, and

$$AV = [Av_1 ... Av_p 0 ... 0] = [\sigma_1 u_1 ... \sigma_p u_p 0 ... 0]$$

If $D$ is the $p \times p$ diagonal matrix with diagonal entries $\sigma_1, ..., \sigma_p$ arranged in decreasing order and if $\Sigma$ is the $M \times N$ matrix defined by

$$\Sigma = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}$$

then

$$U \Sigma = [u_1 u_2 ... u_M] \begin{pmatrix} \sigma_1 & 0 & 0 \\ \vdots & \sigma_p & 0 \\ 0 & 0 & \ddots \end{pmatrix}$$
\[ = [\sigma_1 u_1 \cdots \sigma_p v_p \ 0 \cdots 0] = AV. \]

Since V is orthogonal, we get \( U\Sigma V^T = AVV^T = A. \)

In this decomposition, the columns of \( V \) (rows of \( V^T \)) are called right singular vectors of \( A \) and the columns of \( U \) are called left singular vectors of \( A \).

C. Construction of \( U \) and \( V \)

We summarize here the construction of the matrices \( U \) and \( V \) in the SVD of a matrix.

Let \( A \) be an \( M \times N \) matrix.

- **Step 1:** Orthonormally diagonalize \( A^T A \).
  Compute the eigenvalues of \( A^T A \), \( \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N \), and find a corresponding orthonormal set of eigenvectors \( v_1, \ldots, v_N \).

- **Step 2:** Construct \( \Sigma \) and \( V \).
  For each eigenvalue \( \lambda_i \) of \( A^T A \) compute the \( i^{th} \) singular value of \( A \), \( \sigma_i = \sqrt{\lambda_i} \); these are the entries of the main diagonal of \( \Sigma \), this is \( \Sigma_{ii} = \sigma_i, \ i = 1, 2, \ldots, \min\{M, N\} \), \( \sigma_1 \geq \ldots \geq \sigma_N \). We define \( V = [v_1 \cdots v_N] \).

- **Step 3:** Construct \( U \).
  Assuming \( \text{Rank}(A) = p \), the first \( p \) columns of \( U \) are the normalized vectors \( u_1, \ldots, u_p \) obtained from \( Av_1, \ldots, Av_p \). The additional columns of \( U \) are obtained by extending \( u_1, \ldots, u_p \) to an orthonormal basis of \( \mathbb{R}^{\max\{M,N\}} \).

3.2. **Principal Component Analysis.** Principal component analysis is a practical way of analyzing data, specially in those cases when there is a big amount of data. It can be applied to any data consisting of a list of measurements made on a group of objects or individuals. For instance, consider a group of 30 persons and each of them is subjected to 3 measurements: weight, height, and age. For each person we have a row vector with 3 entries (i.e. a row vector in \( \mathbb{R}^3 \)). In that case the data-set matrix is a \( 30 \times 3 \) matrix and it can be viewed as a set of 30 points in \( \mathbb{R}^3 \). This matrix is sometimes called matrix of observations, where each row represents an individual and each column represent an observation (i.e. height, weight or age). PCA consists, mainly, of computing the eigenvectors and eigenvalues of the covariance of the matrix of observation, but to understand this process we first study some concepts and definitions.

A. Mean and Covariance

Let \( X_1, X_2, \ldots, X_P \) be row vectors in \( \mathbb{R}^Q \) and let \( X \) be the matrix whose rows are \( X_1, X_2, \ldots, X_P \). We think of \( X \) as a matrix of observations, where the \( i^{th} \) row represents the \( i^{th} \) sample and the \( j^{th} \) column represents the \( j^{th} \) measurement. The sample mean is defined as
\[ \bar{m} = \frac{1}{P - 1} \sum_{i=1}^{P} X_i. \]

Thus, the sample mean is the row vector whose \( j \)th entry is the average of the \( j \)th entries of the row vectors \( X_1, X_2, ..., X_P \).

Define \( \bar{X} \) as the \( P \times Q \) matrix whose \( i \)th row is \( \bar{X}_i = X_i - \bar{m} \), \( i = 1, 2, ..., P \). Since \( \bar{X}_1, \bar{X}_2, ..., \bar{X}_P \) have zero sample mean, \( \bar{X} \) is said to be in mean-deviation form.

The sample covariance matrix \( Z_X \) is the \( Q \times Q \) symmetric matrix
\[
\frac{1}{P-1} \bar{X}^T \bar{X}.
\]
We remark that

- For \( j = 1, ..., Q \) the diagonal entry of \( Z_X \) (i.e. \( z_{jj} \)) is the variance of the \( j \)th observation (i.e. the \( j \)th column of \( X \)).
- The sum of all the variances (i.e. the sum of the entries of the main diagonal of \( Z_X \)) is the total variance of the observations, and is denoted \( \text{Var}(X) \)
- The entry \( z_{ij} \) of \( Z \) with \( i \neq j \) is the covariance of the \( i \)th and the \( j \)th columns of \( X \). If the covariance between them is zero we say that they are uncorrelated.

B. Principal components

From this point on we assume that \( X \) is a \( P \times Q \) matrix with rows denoted by \( X_i \), \( i = 1, ..., P \), and that \( X \) is in mean-deviation form. Analysis of multivariate data is greatly simplified when all columns of the covariance matrix are uncorrelated (i.e. \( Z_X \) is a diagonal matrix), thus we want to find a \( Q \times Q \) orthogonal matrix \( U \) with \( X^T = U Y^T \) such that the columns of \( Y \) are uncorrelated and arranged in order of decreasing variance. We can see this as assigning to each vector \( X_i \) a new row vector \( Y_i \) such that \( X_i^T = U Y_i^T \). Note that \( Y_i \) is the coordinate vector of \( X_i \) with respect to the orthogonal basis of \( \mathbb{R}^Q \) given by the columns of \( U \).

Since \( U \) is orthogonal \( X^T = U Y^T \Rightarrow U^T X^T = Y^T \Rightarrow X U = Y \) so the covariance matrix of \( Y \) is
\[
Z_Y = \frac{1}{P-1} Y^T Y = \frac{1}{P-1} (U^T X^T)(XU) = U^T Z_X U.
\]
Hence, we need to find \( U \) such that \( U^T Z_X U \) is a diagonal matrix. If \( U \) is an orthonormal matrix whose columns are given by eigenvectors of \( Z_X \), then \( Z_Y \) turn out diagonal. Since \( Z_X \) is a symmetric, positive semi-definite matrix, it can be orthonormally diagonalized and all its eigenvalues are nonnegative real values. Let \( Z_Y \) be the diagonal matrix whose entries in the diagonal are the eigenvalues of \( Z_X \) arranged in decreasing order (i.e. \( \lambda_1 \geq ... \geq \lambda_Q \)), and \( U \) be a \( Q \times Q \) orthogonal matrix whose columns are corresponding unit eigenvectors \( u_1, ..., u_Q \) (i.e. \( Z_X u_i = \lambda_i u_i \)). We call \( u_i \) the \( i \)th principal component of the data.

The principal components of \( X \), as defined above, are given by an orthonormal basis of eigenvectors of the covariance matrix \( \frac{1}{P-1} X^T X \). From section 3.1, the eigenvectors of \( \frac{1}{P-1} X^T X \) are the right singular vectors of \( X \).
C. Dimensionality Reduction

Note that the orthonormal change of basis $X = YU^T$ does not change the total variance of the data-set $X$: $\text{Var}(X) = \text{Var}(Y) = \lambda_1 + \ldots + \lambda_Q$. For fix $j$ the variance of $\{y_{ij}\}_{i=1}^P$ is $\lambda_j$ and $\frac{\lambda_j}{\text{Var}(X)}$ measures the fraction of the total variance that corresponds to $\{y_{ij}\}_{i=1}^P$. Since $\lambda_1 \geq \ldots \geq \lambda_Q$, $\lambda_Q$ contributes the least to the total variance, so $\lambda_1 + \ldots + \lambda_{Q-1}$ gives a good approximation to it.

Since $X^T = UY^T$, $X_i^T = UY_i^T = [u_1 \ldots u_Q]$ and assuming $\lambda_Q$ is very small (then $y_{iQ}$ is very small since the mean is zero), $X_i$ can be approximated by $X_i^T \approx [u_1 \ldots u_{Q-1}]$.

If the data-set is big enough, it is reasonable to assume that the last few principal components are contributing very little information to the data-set. In that case it is possible to get a good approximation of any observation without considering those principal components. This technique is sometimes used in data compression and, as we will see in Section 4, it is also used in face recognition.

4. Face Recognition Techniques

In this section we will discuss the implementation of PCA and SVD on face recognition techniques. The process here discussed consists of computing the principal components of a database of face images. Then these principal components can be used to approximate face images in and outside the database.

4.1 Matrix of observations

Consider a database containing $M$ face images of size $t \times s$, $I_1, I_2, \ldots, I_M$. For $i = 1, \ldots, M$ the image $I_i$ is a $t \times s$ matrix with columns $C_{i1}, C_{i2}, \ldots, C_{is}$, hence it can be reshaped into a $1 \times ts$ row vector by concatenating its columns; that is $I_i = [C_{i1}^T C_{i2}^T \ldots C_{is}^T]$ and it is reshaped into $\bar{I}_i = [C_{i1}^T C_{i2}^T \ldots C_{is}^T]$.

We often refer to the $j^{th}$ entry of the row vector $\bar{I}_i$ as the $j^{th}$ pixel of the $i^{th}$ image. Let $N = ts$ and let $X$ be the $M \times N$ matrix whose $i^{th}$ row is the row vector $\bar{I}_i$; then $X$ is the matrix of observations, as described in section 3.2, where the image $I_i$ is the $i^{th}$ sample, and the $j^{th}$ column represents the $j^{th}$ measurement (i.e. the intensity value of the $j^{th}$ pixel).
4.2 Principal components of $X$

To the matrix of observations $X$ we can apply the process described in section 3.2 with $P = M$ and $Q = N$.

Let $\bar{m}$ be the mean of the matrix of observations $X$, and $\bar{X}$ be the matrix whose $i^{th}$ row is given by $\bar{X}_i = X_i - \bar{m}$, such that $\bar{X}$ is in mean-deviation form. We compute the eigenvalues of the covariance matrix $Z_X = \frac{1}{M-1} \bar{X}^T \bar{X}$, arrange them in decreasing order $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N \geq 0$ and find an orthonormal set $v_1, ..., v_N$ of corresponding eigenvectors, these are the principal components of our matrix of observations.

4.3 Implementation

The $i^{th}$ principal component, $v_i$, can be reshaped into a $t \times s$ matrix $V_i$ whose entries in the first column are the first $t$ entries of $v_i$, the entries on the second column are the next $t$ entries of $v_i$ and so forth. The matrix $V_i$ is also a face-like image and is called the $i^{th}$ eigenface of the database. Since $\{v_1, ..., v_N\}$ is an
orthonormal basis for $\mathbb{R}^N$, the set $\{V_1, \ldots, V_N\}$ is an orthonormal basis for the vector space of matrices of size $t \times s$. If $A$ is a $t \times s$ matrix we can write $A$ as

$$A = \sum_{i=1}^{N} c_i V_i$$

where $c_i$ is the dot product of $A$ with $V_i$. This is true, in particular, for $t \times s$ face images.

Recall from section 3.2 that the last few principal components are contributing very little information to the total variance of the database. Hence $A$ can be approximated by

$$A \approx \sum_{i=1}^{N-k} c_i V_i$$

where $k$ is an appropriately chosen positive integer.

In applications we might have a database consisting of hundreds of images, but on the other hand these images, even if they are of small size (for instance $128 \times 128$), consists of thousands of pixels; so that in general $N \gg M$. After applying the PCA to $\bar{X}$ we get $N$ eigenfaces (i.e. Principal components) but the last $N-M$ usually contain very little information. Therefore, it is reasonable to only store the first $M$ eigenfaces and use them to approximate any $t \times s$ face image after subtracting the mean of the dataset. More precisely, if $\bar{A}$ is the matrix $A$ after subtracting the mean of the dataset, then

$$\bar{A} \approx \sum_{i=1}^{M} c_j V_j.$$  

Implementation of this method through the use of MATLAB is shown in Section 5.

5. MatLab Implementation

Here we use the method described above and applied to a database of 380 images [1]. The database consist of 10 different images of 38 subjects. Each image is of size $112 \times 92$.

Contents.

- Reading Images
- Converting Images to vectors
- Matrix of Observations $X$
- Writing $X$ in mean-deviation form
- Applying SVD to $X$
- Recovering Eigenfaces
- Approximation of an image in the database
Figure 5. On the top, the first three eigenfaces of the database [1], where we can see the face-like form of this images. In the bottom, the last three eigenfaces of the database; here we can see how these last three eigenfaces contain very little information contributing to the total variance of the database.

- Approximation of an image outside the database

Reading Images.

```matlab
file = dir('C:\Faces\*.pgm');  %Reading the images from the file.
NF = length(file);
images = cell(NF,1);
for k = 1 : NF
    images{k} = im2double(imread(file(k).name));
end
[a b]=size(images{1});  \%a = 112, b = 92
```

Converting Images to vectors.

```matlab
vectorimages = cell(NF,1);
for j=1:NF
```
vectorimages{j}=reshape(images{j},1,a*b);
end %reshape the j^{th} image into a vector as discussed
   %in Section 4.1

Matrix of Observations X.
X=zeros(NF,a*b);
for j=1:NF
   X(j,:) = vectorimages{j}; %Construct the matrix of observations as
end %discussed in Section 4.1

Writing X in mean-deviation form.
M=mean(X);
for i=1:NF
   Y(i,:)=X(i,:)-M; %Subtracting the mean from the matrix of observations
end
X1 =(1/(NF-1)).*Y;

Applying SVD to X1.
[U,S,V]=svd(X1,'econ'); %The command svd creates a matrix U whose columns
   %are the left singular vectors of X1, diagonal matrix
   %S whose entries in the main diagonal are the singular
   %values of X1 arranged in decreasing order and a matrix
   %V whose column are the left singular vectors of X1.
   %Following the explanation given in Section 4.3, the
   %command svd(X1,'econ'), computes only the first NF
   %right singular vectors of X1

Recovering Eigenfaces.
eigenfaces=cell(NF,1);
for i=1:NF
   eigenfaces{i}= reshape(V(:,i),a,b); %Reshaping the left singular vectors
   %of X1 into an a x b matrix
end

Approximation of an image in the database. In this section we use the eigen-
faces to approximate an image that is in the original database
face = im2double(imread('Z.pgm'));
Face = face - reshape(M,a,b);
Faceapprox=zeros(a,b);
for i=1:NF
   Faceapprox = Faceapprox + (sum(sum(Face.*(eigenfaces{i}))))*(eigenfaces{i});
end
Faceapprox = Faceapprox + reshape(M,a,b);

subplot(1,2,1), imshow(face), title('Original image')
subplot(1,2,2), imshow(Faceapprox,[],title('Aproximation'))
Approximation of an image outside the database. In this section we use the eigenfaces to approximate an image that is not in the original database.

```matlab
face = im2double(imread('C:\Test\test.pgm'));
Face = face - reshape(M,a,b);
Faceapprox=zeros(a,b);
for i=1:NF
    Faceapprox = Faceapprox + (sum(sum(Face.*(eigenfaces{i}))))*(eigenfaces{i});
end
Faceapprox = Faceapprox + reshape(M,a,b);
figure
subplot(1,2,1), imshow(face), title('Original image')
subplot(1,2,2), imshow(Faceapprox,[]), title('Aproximation')
```
REFERENCES


FERNANDO ROMAN, DEPARTMENT OF MATHEMATICS, 138 CARDWELL HALL, KANSAS STATE UNIVERSITY, MANHATTAN, KS 66506, USA.
E-mail address: yahdie1@ksu.edu