Amirkabir University of Technology
(Tehran Polytechnic)

Computer Engineering & IT Department

Facial Recognition with SVD

Linear Algebra and Applications course

Supervised by Dr.Ehsan Nazerfard

Taher Akbari
Mehran Taghian

Contents

1 Abstract 2

2 Eigenface for face detection 2

3 SVD Compression 4

4 DCT Compression(Additional Score) 4

5 Facial Recognition with SVD 5

6 Problems 8

7 Handling Instructions 9

8 Cheating Alert 9

1 Abstract

This project has applied theory of linear algebra called "singular value decomposition (SVD)" to digital image processing. Two specific areas of digital image processing are investigated and tested. One is digital image compression, and other is face recognition. SVD method can transform matrix A into product USV^T , which allows us to refactoring a digital image in three matrices. The using of singular values of such refactoring allows us to represent the image with a smaller set of values, which can preserve useful features of the original image, but use less storage space in the memory, and achieve the image compression process. The experiments with different singular value are performed, and the compression result was evaluated by compression ratio and quality measurement. To perform face recognition with SVD, we treated the set of known faces as vectors in a subspace, called "face space", spanned by a small group of "basefaces". The projection of a new image onto the baseface is then compared to the set of known faces to identify the face. All tests and experiments are carried out by using MATLAB and Python as computing environment and programming language.to start, we begin with Eigenface approach.The Eigenface approach is considered by many to be the first working facial recognition technology, and it served as the basis for one of the top commercial face recognition technology products. Eigenface refers to an appearance-based approach to face recognition that seeks to capture the variation in a collection of face images and use this information to encode and compare images of individual faces in a holistic (as opposed to a part or feature-based) manner.For this work, we have shown that by using properties and rules of mathematics, in particular matrices, we can manipulate a set of images (matrices) in such a way as to extract the relevant facial information necessary for detection and recognition.

2 Eigenface for face detection

We start with a set of images, denoted by P_i , called training faces (or data base). For this demonstration we gathered the M=165 images from a group of I=15 individuals. Each individual took D=11 different pictures changing their facial expression for each of the I images. The reason this is a necessary procedure is to help with the later step of finding the Eigenvectors associated with the faces. Because we are using the Eigen face approach we needed to convert our images to grayscale , which is done with a simple code in Matlab or Python.

The second step in our process is to find the average matrix relative to our data base(1) . In this program each image is represented by a matrix M_i. From this we create a summation formula to calculate the average matrix of the data base matrices, which is denoted by ψ . The formula for the average is $\psi = \frac{1}{M}\sum_1^M M_i$.

The image associated to the average matrix is called the average face.

Once we have acquired an average face we can move on to the third step of our procedure which is to**subtract the average face from each of the existing faces in our grayscale data base(2)** . The formula associated with this step is . This process $Q_i = M_i - \psi$ is done to each of the individual. What this step does is that it shows how each of the original individual images differs from the relative average of the set of images. This is a highly important condition necessary for finding the Eigen faces.

After this comes our fourth step,**which is to take each of the Q_i images and find its transpose(3)** . Recall that the transpose of a matrix is the matrix obtained by interchanging the rows and the columns. This operation thus corresponds to a counter clock wise rotation of the image associated to the matrix $Q_i$as well as a 180^o flip of the image.

The best way to see how the transpose affects this image in particular is to focus on the way the hair slants across the forehead. You can see that the image is not simply rotated counterclockwise, but that it also has the 180^o as mentioned previously.

The next step, our fifth step,**is to find the covariant image using the two previous steps(4)**. The covariant image of the set of 165 images is the matrix defined by:

$$C = \frac{1}{M} \sum_{i=1}^{M} Q_i Q_i^T$$

Note that each of the images Q_i and Q_i^T has a squared matrix of size p-by-p which is essential for the computation of matrices. Once we compute our covariant image we can move on to step six of the process and arguably the most important step,**finding the eigenvalues relative to the data base images(5).**

Once we find our eigenvalues we need to**find the eigen vector associated to each value(6)** . For each eigenvalue λ_i $(1 \leq i \leq 6)$, we find an eigenvector . It is important to note that each of the vectors has p coordinates.**In order to find the eigenfaces we write which takes each of the six vectors and uses the first six coordinates from each vector(7)** . We then use this new value of to find our eigenfaces. To calculate the 6 eigenfaces we use the formula

$$F_l = \sum_{k=1}^{M} v_{ik} Q_k$$

These six faces therefore represent the common characteristic features of the faces from the data base. Therefore, one could say that an original face image could be reconstructed by some linear combination of these eigenfaces. Recognition software could then **apply a linear algorithm using these eigenfaces to find whether or not someone is in the original database(8)**.

3 SVD Compression

Image compression can be done using a low-rank approximation on the singlevalue decomposition. The single-value composition takes the form

$$D = U\Sigma V^T$$

where the columns of U and V are the left and right singular vectors, respectively, and Σ) has diagonal entries that are the singular values of D. That is

$$\Sigma = diag(\sigma_1, ..., \sigma_m)$$

The low-rank approximation is then done by zeroing out the values in σ where $r > k$, where r is the row index and k is a real number, and $0 \leq k \leq rank(\Sigma)$. This can be viewed as $\Sigma_k = \Sigma diag(0, ..., \sigma_k, ..., \sigma_m)$. So to find the low-rank approximation for k calculate

$$D_k = U(\Sigma - \Sigma_k)V^t$$

An image can be compressed using the above formula. The lower the value of k, the lower the quality of the compressed image will be and the smaller the size of the image will be. Conversely, the higher the value of k as it approaches the rank of the image matrix, the higher the quality of the image will be and the larger the size will be.

4 DCT Compression(Additional Score)

Compression of an image can also be done via a discrete cosine transformation. This algorithm is used in some of the most widely used image and audio formats, like JPEG and MP3.A DCT transforms an image from the spatial domain to the frequency domain, separating the parts of the image into areas of differing importance. It is similar to a discrete Fourier transformation. The process to get the approximation to the process used for the low-rank approximation. First a DCT is found by the following equation:

$$F(u,v) = (\frac{2}{N})^{\frac{1}{2}}(\frac{2}{M})^{\frac{1}{2}}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\Lambda(i)\Lambda(j)cos[\frac{\pi u}{2N}(2i+1)]cos[\frac{\pi v}{2M}(2j+1)]f(i,j)$$

where

$$\Lambda(i) = \begin{cases} \frac{1}{\sqrt{(2)}} & for \epsilon = 0 \\ 1 & Otherwise \end{cases}$$

Once a DCT D is found, an approximation is found by zeroing out the values in D where r > k, where r is the row index and k is a real number, and $0 \leq k \leq rank(D)$. This can be viewed as $D_k = D - diag(0, ..., D_k, ..., D_m)$. So to find the low-rank approximation for k calculate

$$F^{-1}(D - D_k)$$

As before, a lower value of k represents higher compression and lower quality, while a higher value of k indicates a lower compression but higher quality.

5 Facial Recognition with SVD

In the SVD approach,it treats a set of known faces as vectors in a subspace, called "face space", spanned by a small group of "base-faces". Recognition is performed by projecting a new image onto the face space, and then classifying the face by comparing its coordinates (position) in face space with the coordinates (positions) of known faces.

Assume each face image has $m \times n = M$ pixels, and is represented as an $M \times 1$ column vector if . Then, a 'training set' S with N face images of known individuals forms an M × N matrix:

$$S = [f_1, f_2, ..., f_N] \tag{1}$$

The mean image $\bar{f}$ of set S, is given by:

$$\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f_i \tag{2}$$

Subtracting $\bar{f}$ from the original faces gives:

$$a_i = f_i - \bar{f} \quad , \quad i = 1, 2, \cdots, N \tag{3}$$

This gives another M × N matrix A:

$$A = [a_1, a_2, \cdots, a_N] \tag{4}$$

Assume the rank of A is r, and $r \leq N << M$. It can be proved that A has the following Single Value Decomposition (SVD):

$$A = U \Sigma V^T \tag{5}$$

Here, Σ is an M × N diagonal matrix :

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

For $i = 1, 2, \cdots, N$, σ_i are called Singular Values (SV) of matrix A. It can be proved that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0 \quad and \quad \sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_N = 0 \tag{6}$$

Matrix V is an N × N orthogonal matrix:

$$V = [v_1, v_2, \cdots, v_N] \tag{7}$$

That is the column vectors v_i , for i = 1, 2, ..., N, form an orthonormal set:

$$V_i^T V_j = \delta_{ij} = \begin{cases} 1 & for \quad i = j \\ 0 & for \quad i \neq j \end{cases}$$

Matrix U is an M × M orthogonal matrix:

$$U = [u_1, u_2, \cdots, u_r, u_{r+1}, \cdots, u_M] \tag{8}$$

That is the column vectors u_i , for i = 1, 2, ..., M, also form an orthonormal set:

$$u_i^T u_j = \delta_{ij} = \begin{cases} 1 & for \quad i = j \\ 0 & for \quad i \neq j \end{cases}$$

The v_i's and u_i's are called right and left singular-vectors of A. From (5),

$$AV = U\Sigma$$

We get:

$$Av_i = \delta_{ij} = \begin{cases} \sigma_i u_i & for \quad i = 1, 2, \cdots, r \\ 0 & for \quad i \neq j \end{cases}$$

It can be proved that $u_1, \cdots, u_r$ form an orthonormal basis for R(A), the range (column) subspace of matrix A. Since matrix A is formed from a training set S with N face images, R(A) is called a 'face subspace' in the 'image space' of m × n pixels. Each $u_i, i = 1, 2, \cdots, r$, can be called a 'base-face'.

Let $X(= [x_1, x_2, \cdots, x_r]^T)$ be the coordinates (position) of any m × n face image f in the face subspace. Then it is the scalar projection of $f - \bar{f}$ onto the base-faces:

$$X = [u_1, u_2, \cdots, u_r]^T (f - \bar{f}) \tag{9}$$

This coordinate vector x is used to find which of the training faces best describes the face f. That is to find some training face $f_i, i = 1, 2, \cdots, N$, that minimizes the distance:

$$\epsilon_i = ||x - x_i||_2 = [(x - x_i)^T (x - x_i)]^{1/2} \tag{10}$$

where x_i is the coordinate vector of f_i , which is the scalar projection of $f_i - \bar{f}$ onto the base-faces:

$$x_i = [u_1, \cdots, u_r]^T (f_i - \bar{f}) \tag{11}$$

A face f is classified as face f_i when the minimum ϵ_i is less than some predefined threshold ϵ_0 . Otherwise the face f is classified as "unknown face". If f is not a face, its distance to the face subspace will be greater than 0. Since the vector projection of $f - \bar{f}$ onto the face space is given by

$$f_p = [u_1, u_2, \cdots, u_r] X \tag{12}$$

where x is given in (13). The distance of f to the face space is the distance between $f - \bar{f}$ and the projection f_p onto the face space:

$$\epsilon_f = ||(f - \bar{f}) - f_p||_2 = [(f - \bar{f} - f_p)^T(f - \bar{f} - f_p)]^{1/2} \tag{13}$$

If ϵ_f is greater than some predefined threshold ϵ_1 , then f is not a face image.

Notes:

1) In practice, a smaller number of basefaces than r is sufficient for identification, because accurate reconstruction is not a requirement. This smaller number of significant base-faces is chosen as those with the largest associated singular values.

2). In the training set, each individual could have more than one face images with different angles, expressions, and so on. In this case, we can use the average of them for the identification.

6 Problems

- Implement all the parts which are bold in section 2.
- According to section 3, for $k = 5, \cdots, 105 \quad , \quad k = k{+}25$ Compress 5th image of dataset.And draw compression rate-k chart for SVD.
- (Additional Score)According to section 4, for $k = 5, \cdots, 105 \quad , \quad k = k + 25$ Compress 5th image of dataset.And draw compression rate-k chart for SVD.
- Compare DCT and SVD compression rate and show on a chart.
- Read all images from dataset and randomly choose 10% of them as test set and the remaining as training set.
- Load the whole training set into a matrix called X.
- Calculate the mean of train images and show it.
- Subtract mean image which calculated above from each row of matrix X.
- Perform SVD on matrix X and acquire V^T.The dimension of each row of V^T is equal to dimension of original image. The i_{th} row of V^T known as an i_{th} EigenImage.Showing the first 10 EigenImage.
- We can estimate X with considering first r elements on the diagonal Σ, first r rows of matrix U and first r columns of matrix V and say that approximation

matrix of order r is shown by $\hat{X}_r$. Draw r-order approximation error graph as a function of r.

- The r EigenImages span r dimensional subspace of original image space, By using this r EigenImages, we can generate all the images in train and test. The coefficient of these vectors construct a r dimensional ”Feature Vector”. For each of image in train and test, find feature vector.
- For r = 15 find EigenImage for test and training set and find linear regression by using train to evaluate test set.Draw accuracy chart using test set as a function of r.

7 Handling Instructions

1. dowmload dataset from:`http://vision.ucsd.edu/content/yale-face-database`
2. All your project should be implemented by MATLAB or Python.
3. You are not allowed use any package for implementation of each part.
4. you can use any package For implementation Additional Score part.
5. About 20% of your score would be devoted to the report you deliver within your projects code. In this report you would explain the whole job and anything special you have done.
6. All questions would be answered by the course’s email: alafall18@gmail.com.
7. Place all your modules and report into a .zip named as ”StudentID_FirstName_LastName.zip” before upload.
8. Deadline of the project is at **23:55 pm 97/10/29** .
9. In case of delivery, your code will be downloaded by the responsible TA from Moodle, so the only way to convey your code is Moodle and in if you need to reform your code please upload it when possible to be used in the due date.

8 Cheating Alert

1. This project should be done by individual.
2. Any similarity of more than 30% of the two projects causes the score of both sides to be zero.

Good Luck