Principal Component Analysis for Facial Recognition: Eigenfaces and Singular Value Decomposition

Joe Procopio (003)
Denise Buciuman-Coman (003)
Nicole Dong (002)
Jordan Smart (003)

Abstract

This report addresses a main method behind facial recognition called Principal Component Analysis (PCA). Singular Value Decomposition (SVD) is a branch of PCA and another path to facial recognition. These methods are broken down into finding the covariance matrix and eigenfaces, which are two significant pieces in successfully programing facial recognition. Mathematical derivations of SVD and PCA are used as shown to simplify the methods into more straightforward step-by-step processes that one could perhaps follow and apply. Numerical results include showing an original image of an individual's face and its eigen-reconstruction. MATLAB is used to configure the given image data into an eigenface using SVD. In conclusion, facial recognition is intrinsically a more refined process than face detection because the computer must analyze a changing version of a human face and must be able to consistently identify the person behind the image.

1. Introduction

Perhaps one of the strangest abilities the human possesses is the ability to see a face within inanimate objects. Take for example the face of a human on the surface of Mars as seen in the photo taken by the Viking 1 spacecraft during its orbit of the red planet. This phenomenon is termed pareidolia and is caused by the human tendency to decipher patterns, even in random data, and is not limited to faces. Perhaps more interesting is that this phenomenon does not require a 3D shape for the brain to decipher a pattern. However, computers do not innately have this ability. So the challenge in facial recognition systems is the ability to filter through varying conditions such as lighting, expression, and angle to reach a conclusion in an efficient and accurate manner.

One such trouble regarding efficiency is how a computer analyzes a photo. A photo can be represented as a matrix of values. Assuming the photo to be grey scale and using values between 0-255 to represent the shading of an individual pixel, it can then be quickly seen that a small photo of 640x480 pixels would create a vector containing 300,000 entries of data. A small sample of these photo vectors could then be put into matrix form. The covariance matrix can then be analyzed to find the Eigenfaces that would form the orthonormal basis of the face space. Singular Value Decomposition (SVD) allows the image matrix to be broken into 3 new matrices, two containing eigenvectors, and one containing the positive, non-zero roots of the eigenvalues of the original matrix (its singular values). The SVD produces the basis for the face space from the training set. It is then possible to project test images onto this space and minimize the euclidean distance to identify the face.

2. Mathematical Formulation

2.1 Eigenfaces

Eigenfaces are a significant part behind recognizing a face in real-time. In order to have the ability to locate and track an individual's face in real-time a computer must calculate the most prominent features in a face. The system storing these most significant features, which can include the mouth, eyes, and nose, of a face is called the eigenface. The name involves "eigen" because eigenvectors, also known as the principal components of the image data matrix, are found. These eigenvectors are further transformed into a weighted sum or linear combination of weighted eigenvectors. The weights come from a set of previously chosen faces used as reference data. Eigenfaces are mathematically a linear combination of the weighted eigenvectors and visually a combination of the group of facial reference data. Face recognition comes in when this weighted sum is then compared to the weights of a group of faces previously stored in the system. An important note is that the accuracy and unique comparisons of the faces to be recognized will depend on these initially stored face images.

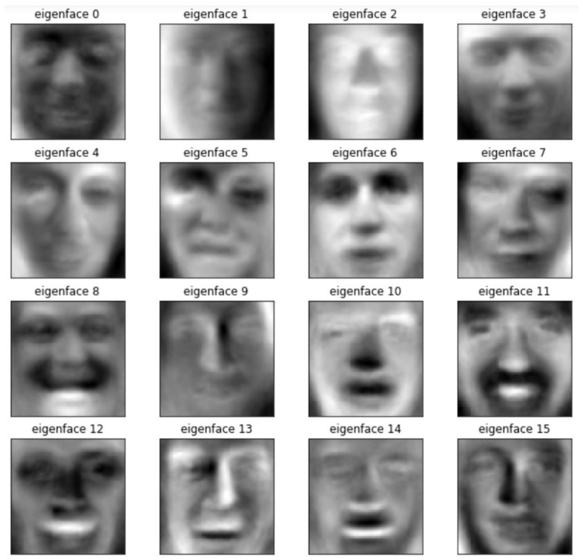

Fig. 1 *Examples of eigenfaces from 16 different individuals.*

2.2 Principal Component Analysis (PCA)

PCA is a statistical approach to facial recognition. This approach uses the construction of eigenfaces to simplify a face into its main features. Using eigenfaces in PCA is known to be one of the most effective methods because of how it reduces images into smaller less data-consuming matrices.

In order to construct an eigenface a few steps are involved, which are also the steps in performing PCA. The first step involves collecting a group database of equal-sized “training faces”. A set, *S*, is created and represented as a vector. These images will be the foundation in building the recognized face into a weighted sum, where the weights in the linear combination are dependent on the training faces. The average face vector is then calculated along using a set of face images.

$$\mathrm{S} = \{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4 \ldots \ldots \Gamma_M\}$$

$$\text{Average face vector} = \Psi' = \frac{1}{M}\sum_{n=1}^{M} \Gamma_n$$

After the average face vector is found it must be subtracted from the original faces and stored into a new vector that will then be used to calculate the covariance matrix, C. [4]

$$\phi_i = \Gamma_i - \Psi'$$

2.3 Covariance Matrix

Eigenfaces are the eigenvectors of the covariance matrix of a set of face images. The covariance matrix is shown below.

$$C = \frac{1}{M}\sum_{n=1}^{M} \phi_n {\phi_n}^T = XX^T$$

X represents the faces in the image matrix. These faces are then compared to the normalized image matrix, $\frac{1}{n}$ X.

$$C = \frac{1}{M}\sum_{j=1}^{M} \frac{1}{a_j a_j^T} = \frac{1}{n} XX^T$$

The product of XX^T is quite large. Therefore, finding the eigenvalues from this product would be data-costly matrix. To solve this, a surrogate of the covariance matrix C is created as $L = A^TA$. The eigenvalues from this new product are found. For an mxn matrix A, both the C and L matrices will have the same number of non-zero eigenvalues, and in fact they will have the same eigenvalues. Using this information solving for the non-zero eigenvalues through the much smaller L matrix is easier for the computer to handle due to the number of images in the training set being much smaller than the pixel information. It is then possible to trim the amount of eigenvalues in the diagonal matrix before computing the eigenvectors (eigenfaces) of the covariance matrix by using only eigenvalues larger than a set value. The larger eigenvalues tend to correspond with the most defining features of a face in the training set. One of the training sets resulted in an eigenface matrix of 36000x19 where one eigenvalue was approximately zero and hence eliminated.

Once L is created, eigenvectors are calculated using the following eigenvalue-eigenvector relationship.

$$AA^T\boldsymbol{v_i} = \mu_i\boldsymbol{v_i} \qquad \begin{array}{l} \mu = eigenvalues \\ \boldsymbol{v} = eigenvectors \end{array}$$

An eigenface is a weighted sum of the eigenvectors. Thus, the eigenface basis, *u*, is found to provide the foundation of the linear combination that will ultimately construct the eigenface. And finally the weights, *w*, from the weighted sum are found and stored in vector form. [4]

$$\boldsymbol{u_i} = \sum_{i=1}^{M} v_i\phi_i$$

$$\boldsymbol{\omega_i} = u_i^T(\Gamma_i - \Psi) = u_i^T\phi_i$$

2.4 Singular Value Decomposition (SVD)

The mathematical formulation for facial detection and facial recognition requires us to compress the file making the larger image into a smaller one. SVD is an approach within PCA but, another path to get similar results. For example, PCA was previously done using a covariance matrix where the covariance was calculated using the product XX^T. SVD involves finding a covariance matrix however, the data matrix X is the decomposition of into A shown below. [7]

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

Decomposing X rather than using the former product XX^T into these three matrices ultimately gives a higher precision to eigenfaces because of how the data is split into three pieces of information. SVD also involves finding eigenvectors, orthonormal matrices, and singular values of a matrix A, which would be the file being compressed. To start, A is defined below where U is an orthonormal matrix of dimension *m x m*. V is an *n x n* orthonormal matrix and Σ is an *m x n* diagonal matrix consisting of positive singular values. SVD must be done in order to simplify and reduce the image matrix to its main features. Following this decomposition of the matrix, the covariance and eigenface is then calculated.

In order to find U, the eigenvectors are calculated from the product of the matrix A and its transpose. Similarly, to find V the opposite product is calculated and the eigenvectors of this new product is found. Σ is the square root of either of those previous products of the matrix A. This mathematical derivation is shown below.

$$K = A^T A$$

$$(K - \lambda I)\boldsymbol{v} = \boldsymbol{0}$$

$$U = \lceil \ \boldsymbol{v_1} \ \mid \ \boldsymbol{v_2} \ \mid \ \dots \boldsymbol{v_n} \ \rceil$$

$$L = AA^T$$

$$(L - \lambda I)\boldsymbol{q} = \boldsymbol{0}$$

$$V = \lceil \ \boldsymbol{q_1} \ \mid \ \boldsymbol{q_2} \ \mid \ \dots \boldsymbol{q_n} \ \rceil$$

$$\Sigma = \sqrt{K} = \sqrt{L}$$

We use the SVD in order to find the eigenvalues and eigenvectors. We find those eigenvalues through Σ which is a diagonal matrix of singular values, and squaring those to reach the eigenvalues. The eigenvalues can then be used to further find the eigenvectors of the covariance matrix.[2] Using the eigenvectors of the covariance matrix (eigenfaces) we get the result as shown below in Examples & Numerical Results.

3. Examples & Numerical Results

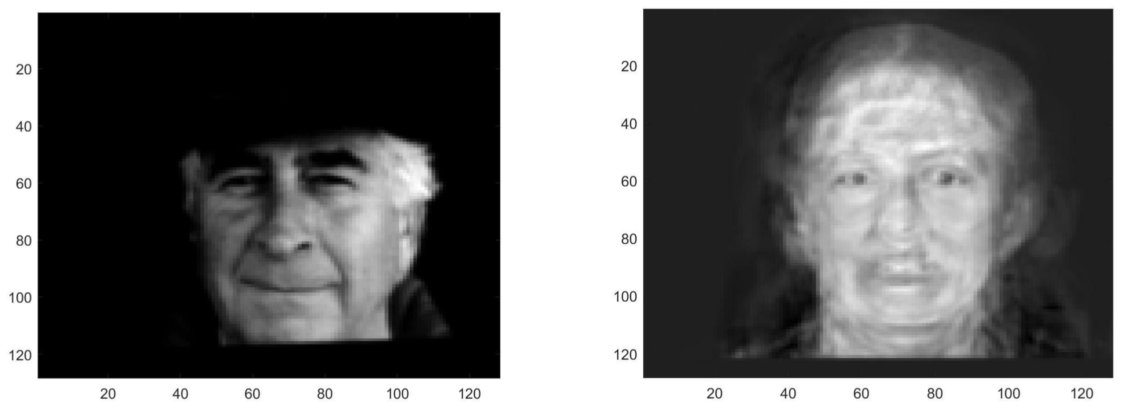

Fig. 2 *Above is an original image (L) e and its eigen reconstruction (R), or projection onto the face space.*

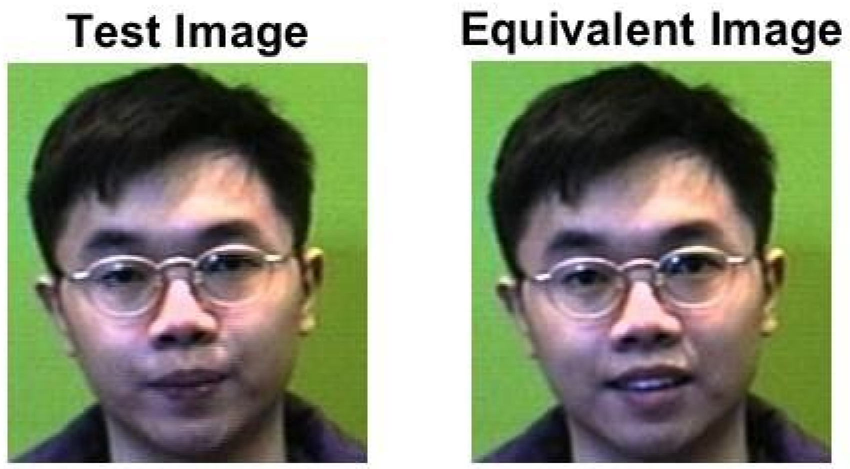

Fig. 3 *Above is the tested image (L) and facial recognition code's matched or "equivalent" image (R).*

MATLAB was used to construct an eigenface, shown in Fig. 2. A set of training data was stored and then loaded within the code. The stored training set was of about 700 images and used to construct the eigenface basis of the eigenvectors (See Appendix B.2). To test whether the MATLAB code works and can recognize a face, the test face image to be recognized is looped into the code. This image is shown on the right in Fig. 3. The code successfully matched the test image with an equivalent image previously stored by showing the resulting match. The match was tested through minimizing the euclidean distance between the actual face and the projection onto the face space. (See Appendix B.1)

4. Discussion & Conclusions

Facial recognition might sound like comparing one image to another and comparing the differences. While we as humans can do this the computer must be programmed to work that precisely. Each picture you take is different to a computer because of expression, lighting, background, glasses, etc. By using SVD and PCA approaches we change the matrix of the image and allow the computer to detect things it can compare (like face shape, distance between eyes, etc) by ignoring the background as well. Using PCA is slightly different from SVD. Normally, PCA requires forming the eigenvectors of the covariance which allows the vectors to be orthonormal. However that can be less precise than using SVD in PCA to find the eigenfaces. By using SVD within PCA we were able to break down the matrix into something more readable to a computer and took significantly less time to compute.

Appendix A: References

[1] Acar, Nev. "Eigenfaces: Recovering Humans from Ghosts." *Medium*, Towards Data Science, 2 Sept. 2018, towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184.

[2] Muller, Neil, et al. "SingularValue Decomposition, Eigenfaces,and 3D Reconstructions." *Drnlm.org*, 2004, drnlm.org/~neil/articles/SIAM_Review_38751.pdf.

[3] O, Alter. "Singular Value Decomposition (SVD) Tutorial." *Singular Value Decomposition (SVD) Tutorial*, 2000, web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm.

[4] Paul, LitonLiton Chandra, and Abdulla Al Sumam. "Face Recognition Using Principal Component Analysis Method ." *Ijarcet*, Nov. 2012, ijarcet.org/wp-content/uploads/IJARCET-VOL-1-ISSUE-9-135-139.pdf.

[5] Sirovich, L, and M Kirby. "Low-Dimensional Procedure for the Characterization of Human Faces." *Engr.case.edu*, 10 Nov. 1989, engr.case.edu/merat_francis/EECS%20490%20F04/References/Face%20Recognition/LD%20Face%20analysis.pdf.

[6] Szummer, Martin. "Face Recognition Project." *Face Recognition Project*, 2002, courses.media.mit.edu/2004fall/mas622j/04.projects/faces/.

[7] Rodrigo de Azevedo 3, et al. "What Is the Intuitive Relationship between SVD and PCA?" *Mathematics Stack Exchange*, 1 Sept. 2010,

math.stackexchange.com/questions/3869/what-is-the-intuitive-relationship-between-svd-and-pca.

Appendix B: Code

B.1

```
% You can customize and fix initial directory paths
TrainDatabasePath = uigetdir('C:\Users\Katsu\Documents\MATLAB', 'Select training database path' );
TestDatabasePath = uigetdir('C:\Users\Katsu\Documents\MATLAB', 'Select test database path');

prompt = {'Enter test image name (a number between 1 to 10):'};
dlg_title = 'Input of PCA-Based Face Recognition System';
num_lines= 1;
def = {'1'};

TestImage  = inputdlg(prompt,dlg_title,num_lines,def);
TestImage = strcat(TestDatabasePath,'\',char(TestImage),'.jpg');
im = imread(TestImage);

T = CreateDatabase(TrainDatabasePath);
[m, A, Eigenfaces] = EigenfaceCore(T);
OutputName = Recognition(TestImage, m, A, Eigenfaces);

SelectedImage = strcat(TrainDatabasePath,'\',OutputName);
SelectedImage = imread(SelectedImage);

imshow(im)
title('Test Image');
figure,imshow(SelectedImage);
title('Equivalent Image');

str = strcat('Matched image is :  ',OutputName);
disp(str)
```

B.2

```
load C:\Users\Katsu\Downloads\faces\faces\ev.mat
load C:\Users\Katsu\Downloads\faces\faces\faceS
load C:\Users\Katsu\Downloads\faces\faces\train_list

figure
```

```
%One can change the file # "xxxx" to any number 1223-5222
fid=fopen('C:\Users\Katsu\Desktop\Matrix Project\rawdata\3084');
Lock = fread(fid);
LockImg = imagesc(reshape(Lock, 128, 128)'); colormap(gray(256));

Elock = zeros(20,1);
for l = 1:20
    for k = 1:20
        v = faceS(k,2:100)';
        i = eigenfaces'*v + mean_face';
        Error = norm(Lock-i);
        Elock(k) = Error;
    end
end

%Displays the eigen projection, i, as an image
figure
Reconstruction = imagesc(reshape(i, 128, 128)'); colormap(gray(256));
```