Face Recognition Using PCA

Applied Maths project

OCHOA Eduardo, VALENCIA Liliana

January 15, 2018

1 Introduction

Face recognition goal is to extract a set of discriminative features from the face images reducing the number of variables so that is one of the most popular application of Principal Component Analysis -PCA-. PCA simplifies the complexity in high dimensional data reducing it by the geometrical projection in low dimensions called *Principal Components* also called *eigenfaces* in the face recognition problem.

The recognition process is done by projecting an entry image in the eigenfaces subspace and it is classify through a comparison with its position in the mention space.

The objective of this project is develop an algorithm that perform face recognition from a given data set of images applying PCA.

2 Methodology

The algorithm to recognize faces have basically two steps: normalization and recognition.

In the normalization process we need to find a transformation that better maps a facial feature into a given location determine by F_i into a 64×64 window. And Recognition which main objective is find the best match between between a input image with a set normalize images.

The fist step is to organize the data, all the .txt files must be in same order and the same characters. A variable f_i is created with the predetermine features.

Features	Coordinates
P1	(13,20)
P2	(50,20)
P3	(34,34)
P4	(16,50)
P5	(48,50)

Table 1: Facial Features Coordinates

2.1 Normalization

Then $\bar{F}$ is initialize with first image of the data set. Two function are created and used to perform the iteration that will give the final $\bar{F}$:

- `Transformation.m`. This function compute the least square to find A and b in the equation

$$f_i^P = Af_i + b \tag{1}$$

  where f_i^P are the predetermine features and f_i are the 5 features of an input image. When the least square is compute, we have 6 resulting values. The first four correspond to A in order to have the matrix $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and the last 2 correspond to b where $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

- `AppyTransformation.m`. This function returns $\bar{F}'$. In order to do it, we give three inputs to the function: A,b and $\bar{F}$ and we apply Eq. 1 to each feature.

In the iteration process we perform the next sequence of steps:

1. Find the transformation matrix A and B for $\bar{F}$ using `Transformation.m`
2. We apply `ApplyTransformation.m` function to find $\bar{F}$ to get $\bar{F}'$. Now we have $\bar{F} \leftarrow \bar{F}'$
3. Using `Tranformation.m` function we find A and b for every image.
4. Finally, we find the average to set $\bar{F}$

We repeat the iteration from step 2 until convergence determine for a threshold of 0.00000001.

With the final $\bar{F}$, we apply `Transformation.m` function to all the images and then map them into a new space of 64×64 and save them to be use in the recognition algorithm.

2.2 Recognition

Before to start the process we divide the normalize images in two data sets: train_images and test_images.For the training set, we use three different images of each person: one frontal view and two side view.

A matrix D with the data set is created where each row is a training image. Also, a matrix L is created with the label of each images (Names). Now, the next steps are compute:

1. Find the mean of the D matrix.
2. Subtract the mean from the D matrix
3. Compute the covariance matrix of D define as

$$\sum = \frac{1}{p-1} D^T D \tag{2}$$

4. Find the Eigenvector and Eigenvalues from the covariance matrix.
5. Compute the principal components -PCA- multiplying the covariance matrix by the Eigenvectors.
6. Compute the feature vectors of all training images multiplying all the images in D matrix by the principal components matrix such that: $\phi_i = X_i \Phi$ where ϕ_i is the feature vector and Φ is the projection matrix that contain the principal components.

To recognize an face, we compute the Euclidean distance between an input image from the test_images data set with all the faces in the train_images data set. Then, we sort the resulting Euclidean distance and the number of images that will be compare to recognize the person will depend on the number k that the user input to find the best match.

3 Description and Results

As we have said the first step of doing a face recognition algorithm is to normalize our set of images into an image space in which we will then apply the PCA process. Thus, the normalization process will result in images with lower dimensions, we have done a transformation from the space of 240x320 pixels to a window of 64x64 pixels. This give us images that we can use to train and test our face recognition program. Some images obtained from the normalization are shown in Figure 1.

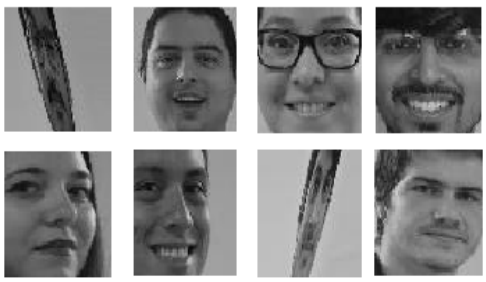

Figure 1: Normalized images in a 64x64 window

We can observe from the Figure that some images have errors when they are transformed into the new window space, this kind of errors result can be generated by errors in the original images coordinates for the 5

features given or from the approximations that are done during the transformation from one space to another. We have observe during our test in the PCA that if the normalization is not correctly done (or at least if it is not done the most appropriate possible) then this will create mistakes in the recognition process, in fact image as the one that look rotated and with a lot of artifacts or in which almost all of the person face is taken away, are those that are difficult to recognize.

The next step was to separate our normalized images into the two sets of images (train and test). For this we took three images of each person into the training set and two of them for the test process. The main idea, as described during the methodology, is to reduced the number of variables in our set of images and project them into the PCA space in where the images has no redundant variables. After this we take each image from the test set and compare how many of them are well recognized with our training images. The accuracy of our face recognition program comes from the equation:

$$accuracy = (1 - \frac{\epsilon}{total\ number\ of\ test\ images}) * 100 \tag{3}$$

The accuracy comes from how many images are well recognized between the two sets and for those that failed we increase a error variable by one, ϵ. This has been tested for different values of k number of faces, in which this k value represents the number of faces (with minimum distance) that our program will look for into the PCA space to find the face. The most important one, obviously, will be the accuracy for a value of $k = 1$, because this value represent that our program found precisely the correct face in the PCA space. In figure 2 we can observe how the accuracy increase while we use greater values of k. For our $k = 1$ we have an accuracy of 53% and we are able to reach an accuracy of 100% just until a value of $k = 94$.

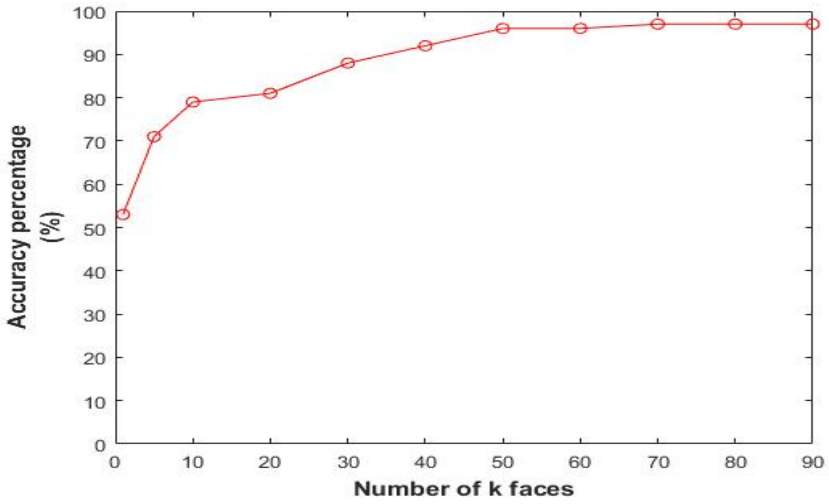

Figure 2: Accuracy values vs Number of k faces

It is important to say that our Face recognition program has a low accuracy when $k = 1$, which show us that the PCA algorithm is dependable on how many good images we use to train the software. So, with a better normalized set of images the accuracy could be improved for the first k face.

3.1 Tests and GUI

A GUI has been created for user purposes and is shown in Figure 3. The main functionalities of this are:

- **Build button:** This button is the one that has to be used to run our PCA algorithm and to load all the variables into the software. It functions as and start button.
- **Image selection menu:** This pop up menu is used to select the image of the training set that you want to recognize.
- **Find:** Button used to try to find the face selected.
- **Edit box (k faces):**Field used to vary the number of faces in which the software will look for the image selected. After changing it you have to rebuild the program.
- **Accuracy field:** Shows the accuracy of the software with the current value of k.

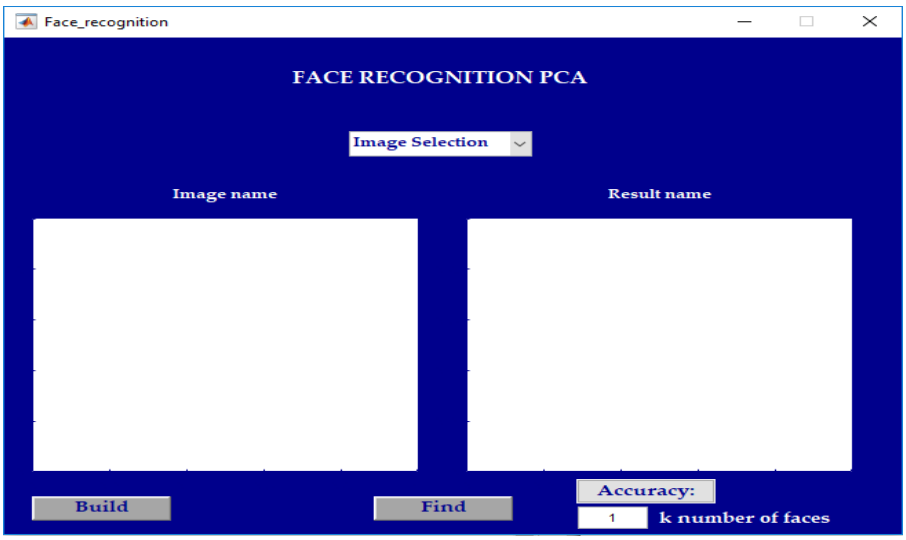

Figure 3: Face recognition GUI

From our tests we show in Figure 4 the GUI working for some images of the test set and the recognized faces for that face. We show in the Figure the accuracy number for two different values of k. In 4c and 4d we can observe how one image that has been tested with a lower value of k and that has not been recognized by the program can be later find by the program if the number of k is raised and that this will produce an accuracy increment too.

Figure 4: Face recognition tests done with GUI implementation

4 Conclusion

Through the development of this project, we improve our understanding about PCA and other topics taken in the applied maths course and its application and real life problems. We also improve our knowledge on finding and understanding errors while programming and determining critical stages of the process since we know that the main issue of our algorithm come from the normalization step and the given data set because if we have good normalize images, the program will be more accurate in the recognizing step.