Principal Component Analysis and Facial Recognition

Diyang Yu, Elif Yurtseven, and Kathryn Mechura

Professor Vinayak Elangovan

8 April, 2017

Outline

Abstract

This project used Principal Component Analysis (PCA) to create a simple facial recognition program. We used faces of people from a variety of gender, age and ethnic groups. The testing images were three: one was the same to one of the training images, one was a side view of one of the training face, and the third was an entirely new face. The program had bo virtual error as it recognized the first testing image as the training image it was supposed to. For the image that was the side view of one of the training images, the program gave two possible training faces that it found it similar to, including the right training face that was the same face facing forward. The new face was matched with one of the training faces, and the error was within our threshold limits, because the match was not correct. Overall, the program does what we expect it to do as it recognizes the testing faces based on size, shape, and the facial features.

Introduction

The goal of this project is to create a program which can successfully recognize the given faces in comparison to the faces that have been previously acquainted with the program. We wanted a minimized margin of error. The project requires two sets of data - one set of training data and one set of testing data. The training dataset consists of ten faces, including two authors' faces, which vary in shape, size, proportions, and age. The testing set includes one face from the training, one author's side face and one new face. To accomplish this we will be using a Principal Component Analysis (PCA) method in the R language. The R language is very useful for this type of mathematical computation involving matrices since there are a lot of built in packages and functions available to make these type of calculations easy. At the end, we hope to

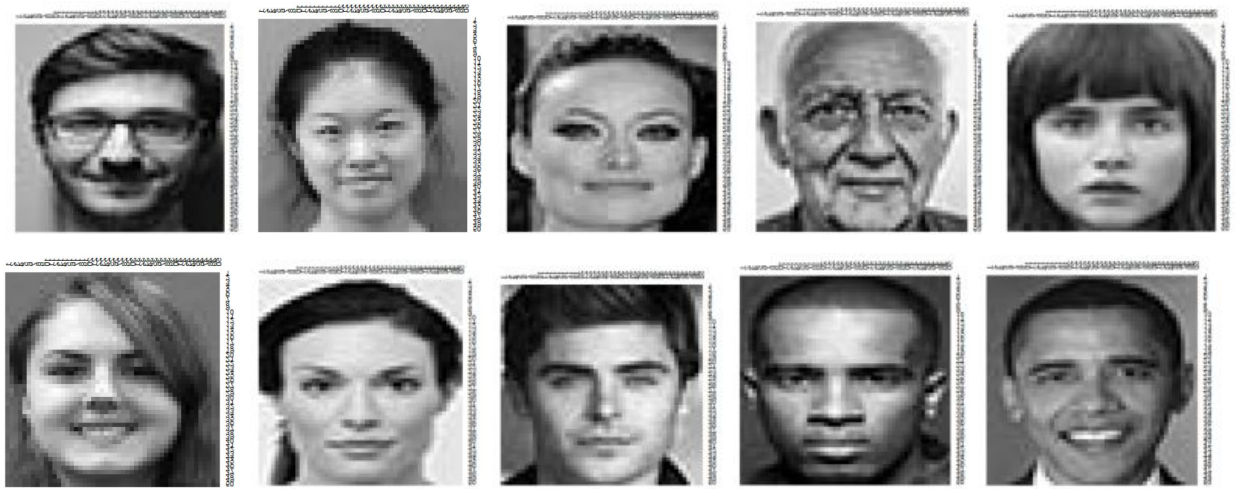
have a program which can identify a testing data when it is one of the training faces facing a different directions. In addition, we hope that we can find a threshold to clarify what is the necessary similarity between a training and a testing data such that the program can tell us that the testing data is absolutely not matching one of training faces.

Theory and Background

Principal Component Analysis (PCA) is a technique for reducing a multi-component dataset into its *k* most important or principal components. The principal components are the ones with the highest influence over the dataset such that when the dataset is reduced to those components, the majority of the data is still correct or intact. PCA uses a covariance matrix and its eigenvalues and eigenvectors, which are derived from the original data. The largest *k* eigenvalues correspond to the principal components of the data and those eigenvalues' eigenvectors are then used to modify the original data into a dataset of smaller dimensions. A presumably easier analysis can then be done on the new dataset as there are many fewer dimensions to account for in the analysis. In our case, the analysis was the attempt to match the testing face to one of the original training faces which is a very simple version of facial recognition. The problem with this kind of facial recognition is that the program assumes that whichever image we out in for testing is a face. That is why even the threshold for matching the faces with the training data is not enough for the program to figure out whether what was inputted for testing is a face or not.

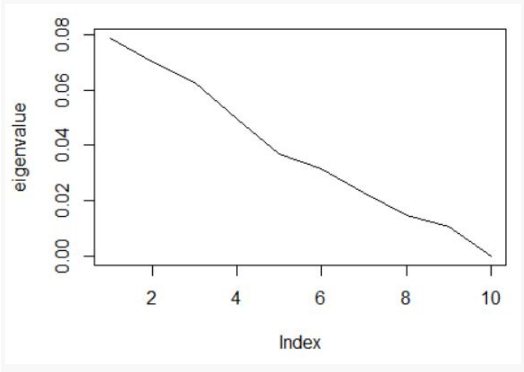
Training Process

First we collected 10 pictures and converted them to 50 by 50 pixels and black and white pictures. Then we read in those pictures and displayed them. Function readJPEG will automatically convert the pictures to matrices. The training faces are shown below.



Then we needed to calculate the mean face and get all feature faces by subtracting the mean from the original faces. Below are all our feature faces. The last image represents the mean face.

After calculating the covariance matrix, the eigenvalues and the eigenvectors using the following functions, we decided to use the first three eigenvalues to get eigenfaces, even though there was no significant difference between eigenvalues,



By multiplying each eigenvalue with each feature face, we got all the eigenfaces. Below are the first best representations of each image. We can see that there is a large amount of information concerning the first eigenvalue. We can already recognize faces comparing them to the eigenfaces. In the code section, we include displays for the second and third eigenvalues, but this did not improve the performance significantly because the first three eigenvalues are too close to each other. At the same time, adding the first three eigenvalues does not improve the performance either.

Testing Process

To test which training face the testing face is closest to, we converted the testing image into a featured face, then projected the face in the eigenspace by multiplying it with each eigenvalue. Then we calculated the Euclidean distance between this eigenface and each eigenface in the training set. The testing image will be recognized as the one where the Euclidean distance is the smallest.

We did three tests. We first picked the second face in the training dataset as a testing data. Theoretically, this can be recognized as a success. After we calculated the Euclidean distance, we found that the second element in the distance vector was the smallest and it was 0, so we matched the testing face with the training face successfully.

```
distance1
##  [1] 0.01712742 0.00000000 0.01871387 0.02240992 0.01657809
0.01354158
##  [7] 0.02367402 0.01612316 0.01622929 0.01073625
which.min(distance1)
## [1] 2
distance1[which.min(distance1)]
```

```
## [1] 0
```

Then, we tested on the second person's face which was facing side. However, the program was not successful at recognizing the same person's face in the training dataset. The Euclidean distance array showed distances to every training face, and the array gave us the closest face as the fourth one.

```
distance2
##  [1] 0.02513939 0.02563349 0.02722946 0.02136007 0.02878651
0.02768686
##  [7] 0.02484743 0.03201081 0.02948645 0.02497370
which.min(distance2)
## [1] 4
distance2[which.min(distance2)]
## [1] 0.02136007
```

The first face above is the testing face, the second one is the original face that is supposed to be recognized as the match to the first (testing) face. The last image is the one which the program recognized as a match. However, the distance from each face was not dramatically different from each other. The difference between the "correct" answer and the real answer was only 0.004. So, if we set the threshold to be smaller than 0.026, we can say that the correct answer is still in our prediction.

Lastly, we picked the third author's face as a testing face, this face had not previously showed up in the training data.

```
distance3
##  [1] 0.01758717 0.01103266 0.01858352 0.02384156 0.01447134
0.01547313
##  [7] 0.02517880 0.01725752 0.01727371 0.01477763
which.min(distance3)
## [1] 2
distance3[which.min(distance3)]
## [1] 0.01103266
```

If we set the threshold as 0.12, then the testing face ends up being closest to the second face in the training data. The distance is even smaller than the previous test where the testing face was one of the same faces in the training dataset. Compared to other faces, the second does seem to be the closest one in size of face and the position of facial features.

Since we are using PCA as a black box, we do not actually know what features it is capturing. It is possible that it is capturing the size of the face and outline, then in the second test, the side face is closest to the 4th face which makes sense because they are similar in size. To make the result more accurate, we can try to increase the resolution of images, which can give us more information.

Conclusion

Overall, the facial recognition using PCA worked relatively well. Although there were some errors identifying the correct face, these could potentially be dismissed if the resolution and the size of the images were increased since there would be more informaiton to include and more

significant differences between faces. Although there are more sophisticated and accurate methods of facial recognition and facial identification, using PCA to do facial recognition is a good method for teaching machine learning techniques to beginners as the concepts are not too difficult to understand.

Team Contributions

Diyang wrote the code, training and test section in the report. Kathryn found, converted, and resized the training and testing faces and wrote the abstract, introduction, background, and conclusion portions of the report. Elif helped with ideas and understanding the code and how PCA works and reviewed and edited important parts of the abstract, introduction, background and conclusion parts of the report.