Eigen Faces: Face Recognition and Analysis

Mini Project in Statistical Methods in AI

Rashika Kheria (201101101)
Student of B.Tech in Computer Science & Engineering
International Institute of Information Technology
Hyderabad, India

Snigdha Agarwal (201101186)
Student of B.Tech in Computer Science & Engineering
International Institute of Information Technology
Hyderabad, India

***Abstract*—we present an approach to identify and verify human faces with the help of eigen faces. Here we try to analyze the results obtained from using PCA and eigen faces on various dataset including IIIT-H's custom created dataset. We also see how eigen faces modifies the image by analyzing the output of running eigen faces algorithm on non-face images.**

I. INTRODUCTION

The face recognition algorithm uses Eigen faces to recognize faces efficiently. This project revolves around eigen faces including their implementation, their efficiency and accuracy, and their effect on images.

The goal of this project was to test PCA using eigen faces on various datasets. The task involved to run both classification as well as verification tests on these algorithms and compare the results.

II. DATASETS

Three dataset were to be used as part of this project:

- Yale Face Dataset
- CMU-PIE Dataset
- SMAI 2013 Students Dataset

A. Yale Dataset

Yale dataset contains images at various angles and elevation of 38 people. We used only those images which have azimuthal angle in the range of -35 to +35 and elevation between -35 to +35. After this filtration, we take 20 images per class which then are resized to 100x100 images. For this filtration and resize, we wrote separate script named extract.sh.

B. CMU-PIE Dataset

In CMU dataset, there are 68 classes each containing 42 images. It is provided to us in matrix format.

C. SMAI 2013 Students Dataset

This dataset contains several images of students and the name of the file denotes class.

This student dataset is trickier as compared to the previous two. Both Yale and CMU dataset are uniform dataset in terms of illumination, angle of the face, background, etc. However, this classroom dataset is full of outliers. Illumination various between images, face alignment is non-uniform, eyes are closed, etc. The main challenge was to achieve good results on this dataset.

III. YALE DATASET RESULTS

For testing the accuracy on this dataset, we used 4-fold method. From the extracted 20 images, 5 images are selected as test data and the remaining becomes train data. We vary these four images among the 20 images and get 4 combinations. We run test for each of class and each combination to calculate the accuracy. We were required to do two tasks on this dataset, Identification using two methods, kNN and SVM and Verification. In order to do identification, we take one fold as test data and try to calculate accuracy on the remaining data as training data. In order to do verification, user is expected to give an image path and a class label (a number between 1-38). In this case, the whole dataset of that particular class is used as training data. If the input image is identified as the same class, then 'Yes, they match.' else "They do not match" message is printed.

The results obtained after running the two classifiers are given below. Also, a detailed analysis is done at the end of it:

A. kNN classification

kNN classifier classification is based on k nearest data points in the training set. We used MATLAB's inbuilt function *knnclassify* for this. The value of k, i.e., the number of nearest neighbors affects the accuracy deeply. We varied *k* from 1 to 9. The result obtained is shown in Table 1. From the Fig.1, it can be deducted that the dataset is very well distributed.

TABLE I. ACCURACY WITH DIFFERENT K VALUES

K - Value	***Accuracy (%)***
2	***100***
3	***100***
5	***100***
7	***100***
9	***99.47***

Accuracy of kNN on Yale Dataset for varies k

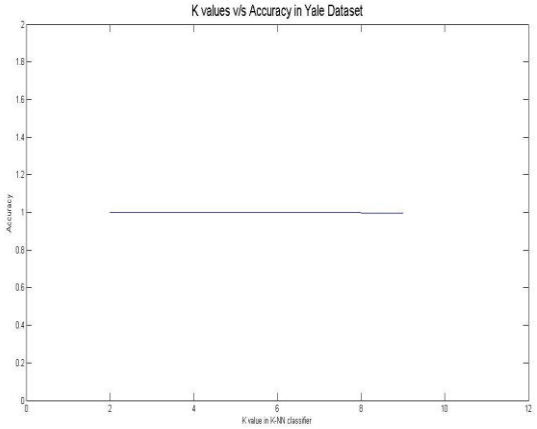

Fig. 1. *Accuracy of Yale Dataset for varying K-values*

B. SVM

Support Vector Machine has the capability to classify even on complex dataset. For this project, we used an open source library *libsvm* in order to use SVM classifier. This library has a function *svmtrain* that takes train labels, weight vector of the training data and other optional parameters. To get the classified class, the function *svmpredict* is called. This takes test probable labels, test weight vector and other options as input.

On applying SVM on this already distributed dataset, the results were expected to be at least the same as kNN. Since the result obtained in kNN was 100%, the result couldn't get better. The result obtained from SVM was also 100%.

C. Results

For this dataset, when we varied *k* in kNN in the first method, we obtained different values (mostly 100%). When number of Eigen vectors were varied in any of the above method, we obtained different accuracies for these values. Number of Eigen vectors was varied from 5, 10, 15, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90 and 100 and we received the following result.

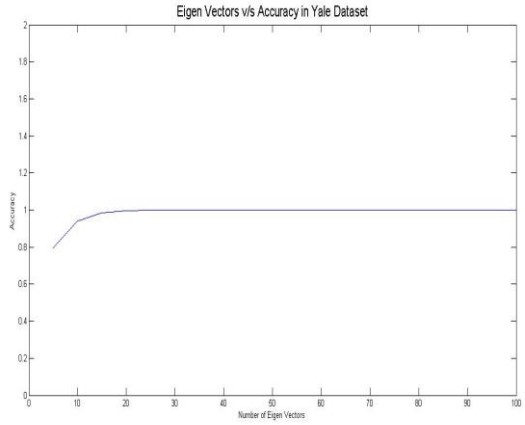

Fig. 2. *Graph showing accuracy variation with different number of eigen vectors*

TABLE II: ACCURACY WITH DIFFERENT # OF EIGEN VECTORS

# of Eigen Vectors	*Accuracy (%)*
5	***79.47***
10	***94.21***
15	***98.42***
20	***99.47***
25	***100***
30	***99.9***
35	***99.9***
40	***99.9***
45	***99.9***
50	***99.9***
60	***99.9***
70	***99.9***
80	***99.9***
90	***100***
100	***100***

Accuracy of kNN on Yale Dataset for different # of Eigen Vectors

These results indicates that Yale dataset is highly distributed and effects of parameters such as *k* or the number of Eigen vectors in PCA does not affect its accuracy much. The results of verification is also similar. For verification, we took two images of the same class as a pair and two images of different class as a pair. 988 such pairs known as imposter pairs were made to calculate threshold for verification purpose. We calculated Euclidean distance between the image of each pair and plotted the ROC curve. The whole dataset was taken as training data and Eigen vectors were calculated. The Euclidean distance between the image weight vector and the weight vector of the class against which we need to verify was calculated. If atleast 3 values were below threshold then the given image was said to belong to the given class label. The result obtained was 75%.

The threshold value used for verification was scaled to a factor of 0.1 before comparing to Euclidean distance. This ensured least False Positive Rate and highest True Positive Rate. ROC for this is shown in Fig. 3.

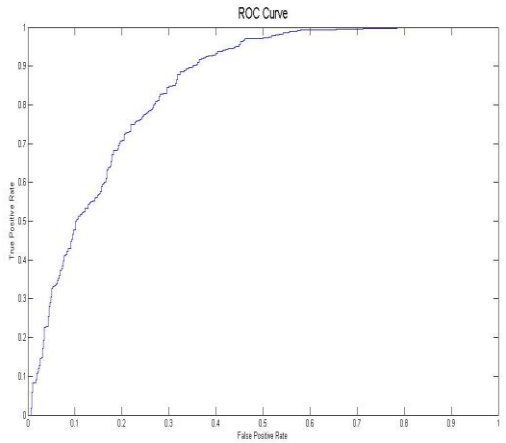

Fig. 3. The ROC curve plotted using distance between Face weights of both the images which was calculated using the whole dataset as training data

IV. CMU-PIE Dataset Results

Here again we used 6-fold method. With 42 images in each class, total 6 pairs is formed and testing against the rest of the dataset as training data. The task was to do both classification and verification for this dataset, same as Yale dataset. The procedure for verification is the same as that of Yale dataset. For identification, following methods were used, kNN and SVM.

Verification was done the same way as it was done for Yale Dataset. Here also, the user is expected to give an image name and identification class number (a number between 1 & 68). The whole dataset of that particular class is used as training data. If the input image is identified as the same class, then 'Yes, they match.' else "They do not match" message is printed.

The results obtained after running the two classifiers are given below. Also, a detailed analysis is done at the end of it:

A. kNN

We used *knnclassify* with varying *k* values. Again, since the data was already well distributed, we obtained high accuracy. The k value was varied from 2 to 9. The result obtained is shown in Table 1.

TABLE III: Accuracy with Different K Values

K - Value	*Accuracy (%)*
2	***100***
3	***100***
5	***100***
7	***100***
9	***100***

Accuracy of kNN on CMU-PIE Dataset for varies k

From the Fig.4, it can be deducted that the dataset is very well distributed.

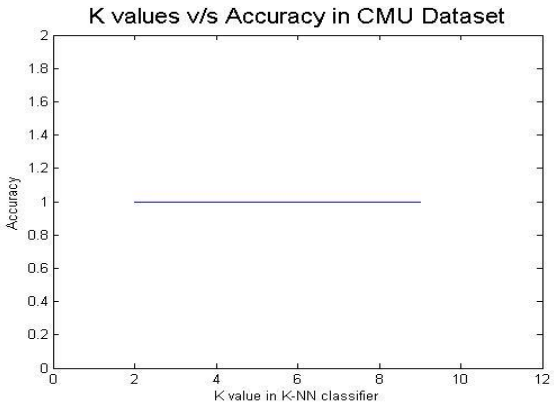

Fig. 4. Accuracy of CMU-PIE data with various k's

Number of Eigen vectors was also varied and the graph obtained is shown in Fig. 5.

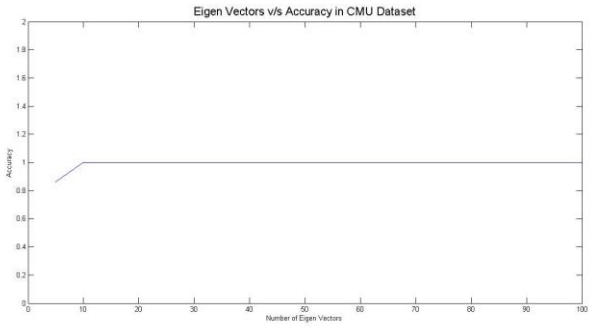

Fig. 5. *Graph showing accuracy v/s Number of Eigen Vectors for CMU Dataset*

TABLE IV: Accuracy with Different #of Eigen Vectors

# of Eigen Vectors	*Accuracy (%)*
5	***86.13***
10	***100***
15	***100***
20	***100***
25	***100***
30	***100***
35	***100***
40	***100***
45	***100***
50	***100***
60	***100***
70	***100***
80	***100***
90	***100***
100	***100***

Accuracy of kNN on CMU-PIE Dataset for different # of Eigen Vectors

B. SVM

We applyied SVM on the already distributed dataset. The percentage of accuracy obtained was the same as kNN. The ROC curve for CMU-PIE dataset is as shown in Fig.6.

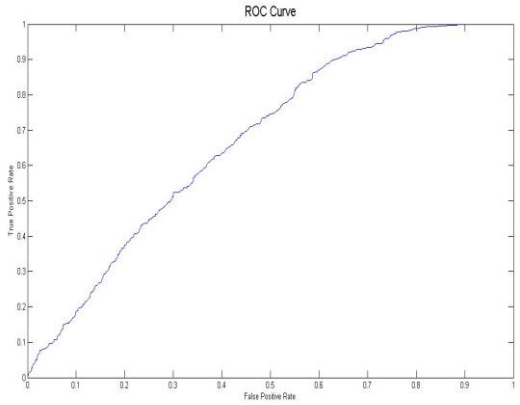

Fig. 6. *ROC Curve*

V. STUDENTS DATASET RESULTS

This dataset was most challenging. All the previous dataset was very well distributed, was in accordance to the requirements of PCA and all images of same class had similar parameters. However, this dataset has lot of intra class variations and outliers. We then filtered the data in such a way that only those classes remain which has 5 images.

A. kNN

We used knnclassify on this dataset with k = 2, and obtained an accuracy of 45.71%. For this we are using 20 Eigen vectors. On decreasing number of Eigen vectors, this accuracy decreased. When the number of Eigen Vectors is 10, the accuracy comes approximately 34.28%.

B. SVM

SVM produced the exact same result.

In order to increase accuracy for this dataset, we can look forward to image processing techniques. Histogram Equalization can help us to solve the problem of varying illumination. In order to nullify other factors that can effect accuracy, we can involve more image processing techniques. Another thing we can do is that we can increase number of images in each class.

VI. PLAYING WITH PCA AND EIGEN FACES

Eigen faces does an amazing job when it comes to recognize a face. However, what happens if we give a non-face image to it? We tried some experiments and found weird results. Every image we gave, after re-construction started looking similar to faces. This thing increased as we increased the number of Eigen vectors. It even happened that after certain number of Eigen vectors, it added spectacles and mustaches on the face. If the image given as input does not contain full area covered with face, the output image contains an image with face expanded to occupy full image.

Some examples of this behavior is shown below:

Fig. 7. *(From left to right) Input image to PCA, reconstruction using 10 Eigen Vectors and reconstruction using 100 Eigen Vectors.*

As we can see, the input image was just a lamp and the output image came is more like a face. As we increased the number of eigen vectors, the original image data started to dominate but it also shows that even at that point, it still looks like a face. If observed carefully, we can see a hint of spectacles of the area which is similar to eyes.

Fig. 8. *(From left to right) Input image to PCA, reconstruction using 10 Eigen Vectors and reconstruction using 100 Eigen Vectors.*

In the above figure, both the output images show no similarity with the input image.

Fig. 9. The image on the left is the input image and the right one is the recontruction using 500 Eigen Vectors

Fig. 9 shows that a totally different image is given a face implantation on it. Since, the number of Eigen vectors taken is very high, original information dominants. This is done to show that the face implantation is nowhere close to the original image.

CONCLUSION

Eigen Faces is a very good technique to recognize faces and this is evident from the accuracy achieved. However, this is true only when the dataset is uniform and have similar parameters in terms of illumination, mood, orientation of face, etc. It is robust to all local changes and this affects its accuracy negatively. However, it is resistant to global changes as those changes will be nullified while we subtract mean from the image.

This also shows that Eigen faces over fit test images to look like faces. Moreover, if an unknown image comes, it transform it to look similar to training set images. Like introducing spectacles, mustaches, different hair style, etc.