ENEE- 439M – Foundations of Machine Learning

Project 1 – Face Recognition using KNN,PCA

Rajeshwar N S

OVERVIEW:

Here, We will implement classifiers like K-Nearest Neighbours and dimensionality reduction techniques like PCA to perform face recognition. We use the given Illumination dataset (Illumination. Mat) which contains images of dimensions (40x48) ie. (1920 pixels each).

PREPROCESSING:

First, we convert the given .mat file to Images by reshaping into (40x48) pixels and use array to Img function and store in folders for easier accessibility of training and test data. Given that, the .mat file consists of 68 faces each having about 21 illuminations. So, we save the data in such a way that there are 68 parent directories each consisting of its corresponding 21 illuminations. Also, we save in a format such that its easier to label the training data. For eg. Face10_illumn10 corresponds to 10th image in face10 directory. The label for this will be 'Face10' which will result if corresponding test image is given. The data is split as 70-30 for train and test set respectively. The average of Images are computed using mean image function for centring which is required for performing PCA.

K-NEAREST NEIGHBOURS:[1]

For implementing the KNN classifier,

1. Calculate the Euclidian distance
2. Get the nearest Neighbours
3. Make Predictions by gathering majority votes

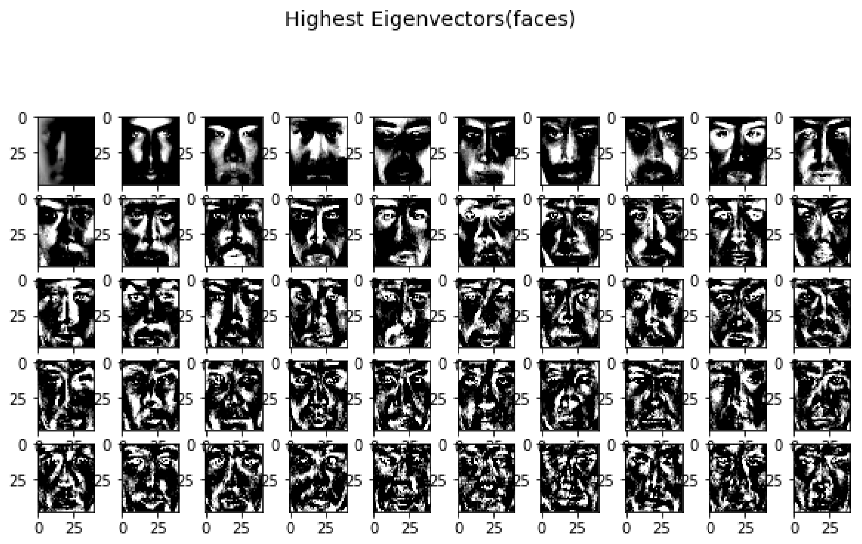
After calculating the distance, To get the nearest neighbours, sort all the Euclidian distances of the new data points to test point. Select the top k for nearest neighbours.

Observation: For a given value of k=1, got an accuracy of around 85% . However, for increasing values of K, the accuracy didn’t increase much in our case. Also, took time of about more than 15 mins on i3-4th Gen CPU.

PRINCIPLE OF COMPONENT ANALYSIS:

For PCA,

1. Calculate Co-variance of mean centred data using np.cov
2. Calculate Eigen values and Eigen Vectors using scipy.linalg
3. Sort the Eigenvalues in decreasing order and compute the highest eigen vector having largest value.
4. Choose the k eigen vectors that correspond to the top k eigenvalues.
5. Project the data points to Eigen vectors to obtain k-dimensional feature space.



Observation: Initially, First top 10 Eigen vectors were used and accuracy turned out to be less compared to to just using KNN without using reduction. However, selecting next subsequent range of eigen vectors seemed to give a better result(>80). Also, most importantly, training with reduced data took only a second or two (for the same k value used before with only KNN) even on an old CPU. So, the computation using dimension reduced is much faster compared to processing the raw high dimensional data

Conclusion:

Classifiers like KNN are discriminative classifiers which gives a good result for finding similarities between data points. However, if the k values are large the classifier may tend to perform on training but predicts incorrectly for test data. And computationally, the performance for this classifier is not great compared to Bayes as it involves the squared distance. This can be overcome by dimensionality reduction techniques like PCA.

REFERENCES:

[1]: https://machinelearningmastery.com **–** KNN Tutorial

[2]: StatQuest with Josh Starmer – PCA- step by step tutorial