EE5907 CA2
Face Recognition

Ma Junting

A0144294A

Project Overview

In this project, the aim is to implement different models and algorithms in order to construct a face recognition system. Principle Component Analysis (PCA) is first implemented to reduce dimensionality of data and to visualize the resultant data. Linear Discriminative Analysis is also implemented perform dimensionality reduction and visualization similar to PCA case. Afterwards, Support Vector Machine (SVM) and Convolutional Neural Network (CNN) are used to classify the face images.

The image data used are from CMU PIE dataset as well as 10 self-images. Out of 68 subjects in the PIE dataset, 20 of them are selected with a random number generator to ensure the randomness of data and prevent bias in the data selection process. 70% of images from each subject are used for training, and 30% of images are used for testing.

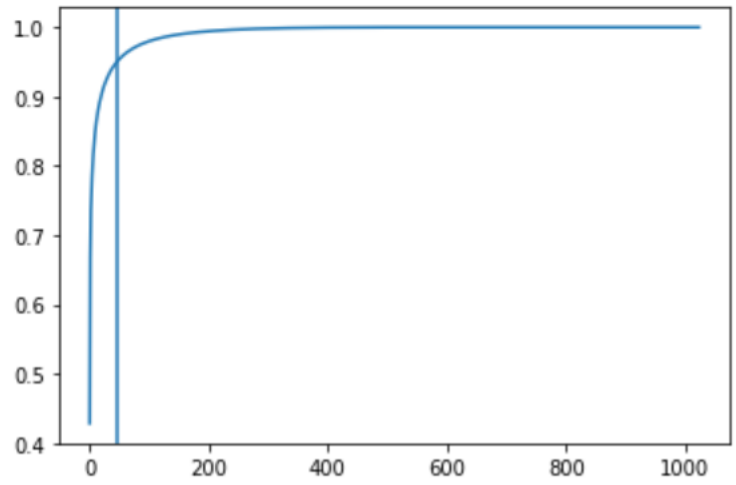
The programs are written in python and in the format of python notebook for ease of troubleshooting and visualization.

PCA for Feature Extraction, Visualization and Classification

For PCA implementation, 500 images from the training data set are randomly sampled and used as the training set together with 7 random images from self-images. The mean of training data is first computed, and the difference between the mean and the training data set is obtained. After obtaining the difference, the covariance is computed, and eigenvalues and eigenvectors of the training set are calculated using ‘numpy’ package. The eigenvalues and eigenvectors are sorted in descending order.

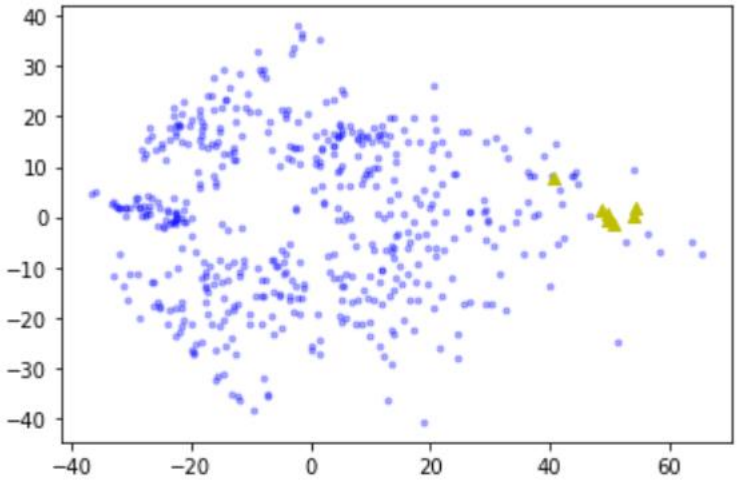
Upon computation of eigenvalues and eigenvectors, the number of features needed to reach 95% threshold can be calculated by finding the minimum number of eigenvalues whose sum is greater than 95% of the total sum of all eigenvalues. The plot of the result is shown below.

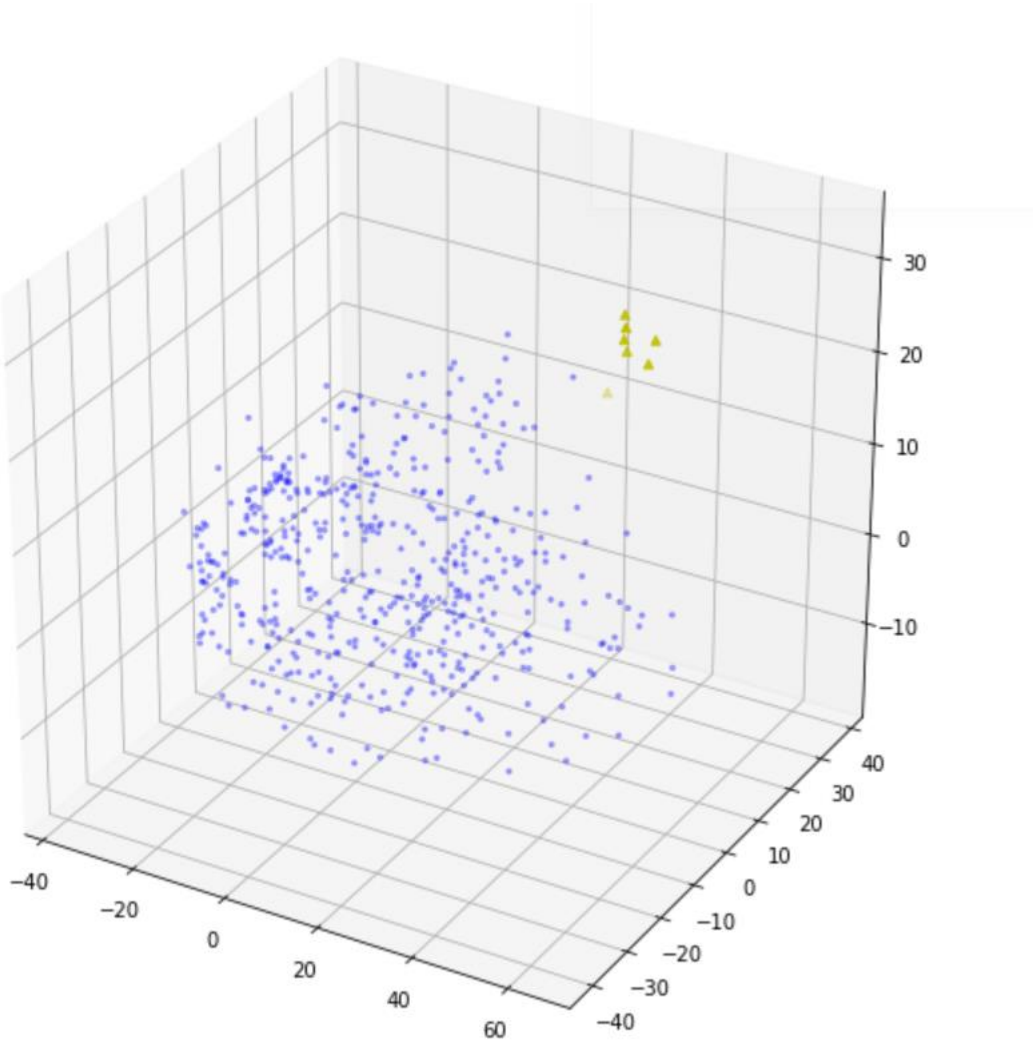
```
include 46 features for 95% threshold
```



Data Visualization

The following plots are the data of training set with dimensionality 2 and 3.





In the two plots, data from PIE dataset are represented by the blue dots, and data from self-

images are represented by the yellow triangles. From the two plots, the training data points spread relatively evenly within the entire data space, while the data points of self-images are very close together. This is possibly because the self-images are taken at the same time with similar background and almost identical surrounding environment such as lighting. This can also explain the reason for the strangely high classification accuracy compared to that of the PIE data set which is to be discussed in the next section.

Classification with K-Nearest Neighbor (KNN) Classifier

KNN classifier is implemented to classify the test images and output classification accuracies. Euclidean distance between each test image and the training data set is computed, and the class that has the smallest Euclidean distance is considered as the class which the test image belongs to.

Classification accuracy is computed for dimensionality 40, 80 and 200, for PIE images and self-images. The results are shown below.

Dimensionality = 40

Classification Accuracy on CMU PIE test images: 62.254901960784316%

Classification Accuracy on own images: 100.0%

Dimensionality = 80

Classification Accuracy on CMU PIE test images: 66.07843137254902%

Classification Accuracy on own images: 100.0%

Dimensionality = 200

Classification Accuracy on CMU PIE test images: 68.13725490196079%

Classification Accuracy on own images: 100.0%

From the classification accuracy data obtained, an increasing trend can be observed for PIE test images. The classification accuracies are generally low, and the possible reason could be that the training set is relatively small (about 24% of entire training set), causing the error rate to be higher than expected. In addition, for KNN classifier, the parameter K is set to 1 by default, which might lead to some degrees of misclassification and contribute to the high error rate.

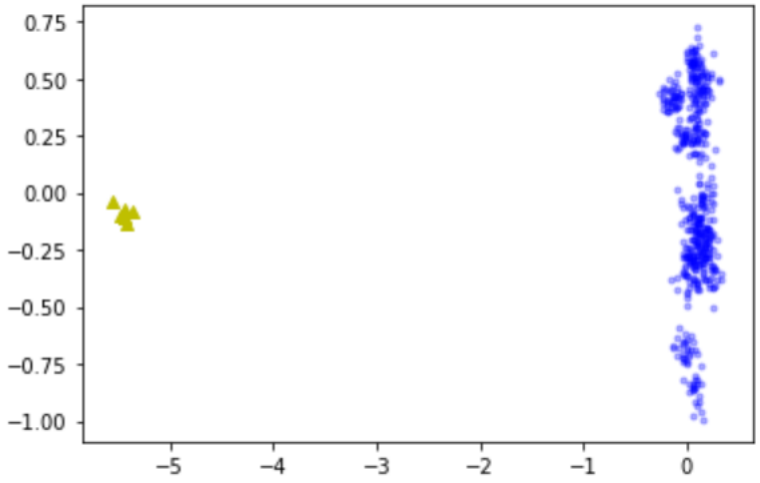
For self-images, as mentioned in the previous section, the images are almost identical to each other, so that it is more likely for the test images to be classified correctly compared to the other PIE test images. In addition, there are only 3 self-images in the test set, and the probability of getting 100% classification accuracy is very high compared to the others.

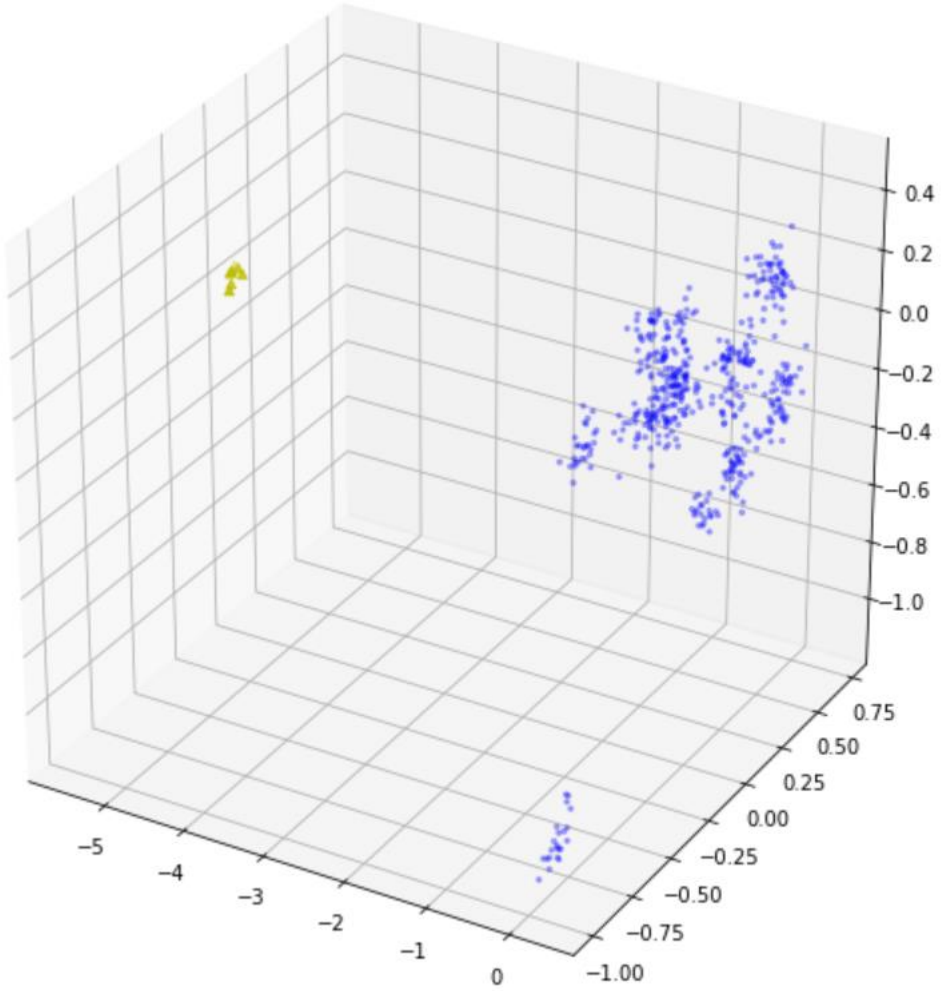
LDA based Data Distribution Visualization and Classification

Similar to PCA, the goal of LDA is to obtain the eigenvalues and eigenvectors of the training set. 500 images are sampled randomly from training images and constructs the training set together with 7 self-images. After preprocessing, scatter within class and scatter between classes are computed, and eigenvalues and eigenvectors are calculated based on scatter within class and scatter between classes.

For dimensionality reduction, the procedure is the same as that in PCA implementation and will not be discussed here. Training and test sets have their dimensionality reduced to 2, 3 and 9 for data distribution visualization and classification.

Data Visualization





The two plots above shows 2D and 3D data distribution for LDA. It can be seen that self-images are very far apart from all the other PIE images. This could be because of the low

eigenvalues. Eigenfaces are attempted to be plotted, and the result is more similar to a group of random noise rather than a human face. It is possible for PIE images to have some similar background noise in the images which cannot be captured by human eyes but identified by LDA. For self-images, they are newly taken and it is likely for them not to have similar noises as compared to the others.

Moreover, clustering can be observed in both plots, especially in 3D plot, where some data points are very close to each other but far apart from all the other points. The points within a cluster is likely to belong to the same class. In this project, up to 3 dimensions are shown. If more dimensions can be observed, the clustering would be more obvious.

Classification with KNN Classifier

The classification is done with KNN classifier similar to PCA, and is done for data with dimensionality of 2, 3 and 9. The results are shown below.

Dimensionality = 2

Classification Accuracy on CMU PIE test images: 20.392156862745097%

Classification Accuracy on own images: 100.0%

Dimensionality = 3

Classification Accuracy on CMU PIE test images: 33.43137254901961%

Classification Accuracy on own images: 100.0%

Dimensionality = 9

Classification Accuracy on CMU PIE test images: 83.52941176470588%

Classification Accuracy on own images: 100.0%

An increasing trend can be observed in the classification accuracies. Compared to PCA, the rate of increase in classification accuracy is much higher. This can be deduced from the observation in the data distribution plots, where clusters are spotted. For self-images, the accuracy is very high. This is reasonable as the images are far apart from all the other images even in the 2D plot.

SVM classification

For SVM classification, raw data as well as preprocessed data using PCA are used. Since it is not specified in the instructions, the entire training set is used, instead of sampling 500 images from the entire training set. Linear SVM is used with different penalty parameters (0.01, 0.1, 1). In SVM, training data might be misclassified, a higher penalty parameter will choose a smaller-margin hyperplane in order to get all training points classified correctly. In another word, penalty parameter controls the tradeoff between lower classification error in training data and lower classification error in testing data. A low penalty parameter results in a higher classification error in training data compared to a higher penalty parameter, but the classification error in testing data can be reduced. If the penalty parameter is too low, under-fitting is likely to happen. On the other hand, if the penalty parameter is too high, over-fitting can occur, resulting in higher classification error in testing data. It is possible to tune the penalty parameter in order for SVM model to provide more accurate prediction without under-fitting or over-fitting.

Classification Result

Raw face images

training accuracy for penalty param = 0.01: 100.0%
test accuracy for penalty param = 0.01: 98.24046920821115%
training accuracy for penalty param = 0.1: 100.0%
test accuracy for penalty param = 0.1: 98.53372434017595%
training accuracy for penalty param = 1: 100.0%
test accuracy for penalty param = 1: 98.63147605083088%

The results above shows the classification accuracy for raw face images. Since the dimensionality of the images is not reduced, all information is retained, and the classification accuracy for training data remains at 100% regardless of the change in penalty parameter. For test data, the classification accuracy keeps increasing as the penalty parameter increases, which contradicts to the theoretical analysis of the effect of penalty parameter. This is possibly to be because that the high amount of information in the data causes different classes to be more distinct from each other, and over-fitting does not occur even when the penalty parameter is 1. On the other hand, the difference in classification accuracy is quite small (less than 0.1%) for penalty parameter of 0.1 and 1, and it could simply because the classification accuracy for penalty parameter of 1 is higher by chance.

Data with dimensionality of 80

training accuracy for penalty param = 0.01: 98.11478843736909%
test accuracy for penalty param = 0.01: 96.57869012707722%
training accuracy for penalty param = 0.1: 99.66485127775451%
test accuracy for penalty param = 0.1: 97.5562072336266%
training accuracy for penalty param = 1: 99.95810640971932%
test accuracy for penalty param = 1: 97.0674486803519%

From the results above, it is obvious that classification accuracy for training data increases as penalty parameter increases, and classification accuracy for testing data increases when penalty parameter is low and classification accuracy decreases when penalty parameter is high. This implies that under-fitting occurs when penalty parameter equals to 0.01, and over-fitting occurs when penalty parameter equals to 1. In this case, the optimum penalty parameter is around 0.1.

Data with dimensionality of 200

training accuracy for penalty param = 0.01: 99.66485127775451%
test accuracy for penalty param = 0.01: 98.1427174975562%
training accuracy for penalty param = 0.1: 100.0%
test accuracy for penalty param = 0.1: 98.72922776148583%
training accuracy for penalty param = 1: 100.0%
test accuracy for penalty param = 1: 98.63147605083088%

The trends of classification accuracy for both training and test data are the same as that for data with dimensionality of 80. The optimum penalty parameter is approximately 0.1. With higher dimensionality, the data retain greater amount of information compared to data with dimensionality of 80. This difference in information leads to a higher classification accuracy in both training and test data.

CNN Classification

A convolutional neural network is constructed with the specifications given in the instructions.

The classification accuracies and losses for training and test data are shown below.

Test data accuracy:97.75171279907227%. Loss: 0.09190742222943643

Training data accuracy:100.0%. Loss: 0.0023320119163828304