PRML PA 3

Perceptron and PCA

Google Colaboratory

Rhythm Patni

(B22CS043)

Question 1:

Implementation of Perceptron Learning Algorithm on a locally generated synthetic dataset.

Task 0 : Data Generation

The dataset generated has 5000 data points, generated randomly using random functions in NumPy. The dataset generated in the colab file.

The data.txt file generated is submitted in the zip file.

The dataset is made linearly separable by assuming the weights randomly and classifying accordingly.

Task 1 : Implementation of perceptron training algorithm

The B22CS043_train.py file submitted contains the implementation of the perceptron training algorithm.

Normalization of the data is done using Min-Max normalization.

The weights generated after all the epochs are stored in a weights.txt file in the same directory.

The training file takes the train.txt file as the argument.

The train.txt file contains 70% of the initial data generated (with labels)

Task 2 : Testing and Evaluation.

The B22CS043_test.py file takes the B22CS043_test.txt file as an argument which contains the remaining 30% of the data without labels. The python script predicts the labels of the data_points and prints them. The training file creates a weights.txt file in the same directory, and stores the predicted weights in that file. The same file is read by the test.txt function and uses the weights to predict the labels of the test set.

Task 3 : Comparison of the results.

In the Colab file, training is done 3 times, using 20%, 50%, 70% of the training data respectively. The results obtained are tabulated as follows:

The train_data function in the colab is responsible for training the dataset. Using split argument. The same function is called using different split values and Accuracy is calculated using the accuracy function defined in the colab file.

S.No.	Percentage of training data	Accuracy (%)
1.	20	99.65
2.	50	99.44
3.	70	99.73

The testing data used is the data not included in the training data. So, the testset for the 20%training data will contain the remaining 80% of the data.

Since the data generated was linearly separable, there will always exist a line which divides the data completely. So even after normalization, it is possible to reach such weights which accurately classifies the data.

Question 2: PCA

Using PCA to reduce the dimension for a face-recognition task.

Task 1 : Loading and Pre-Processing.

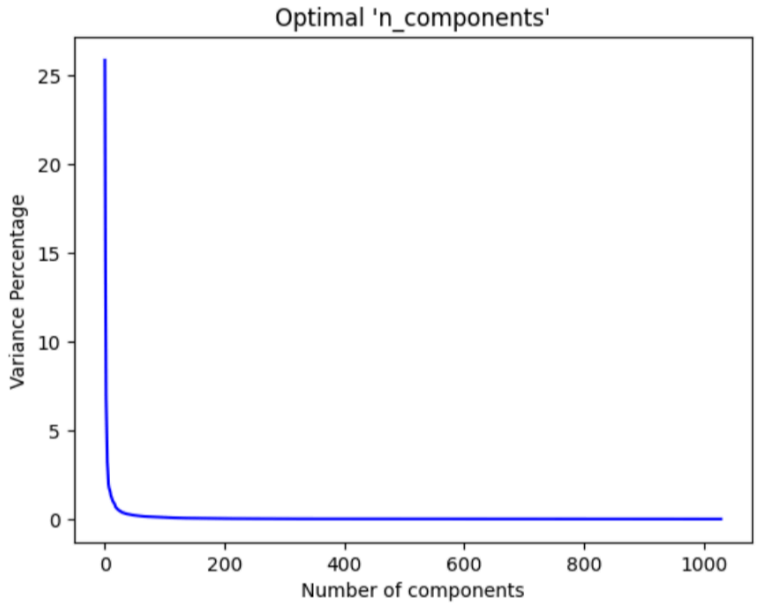
The data has been loaded using the pre-loaded datasets, in the “sklearn” library.

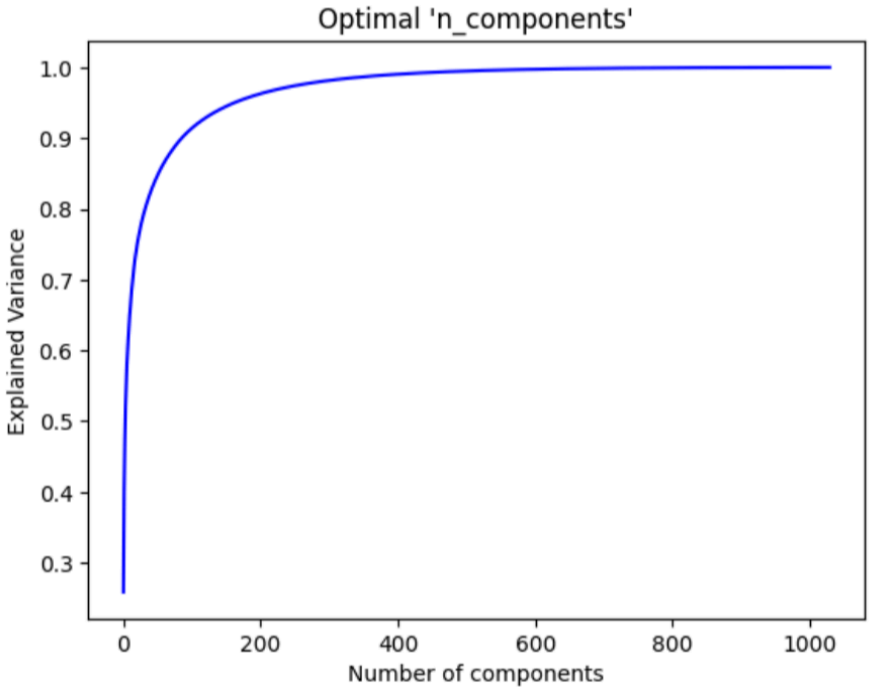
The data has been loaded into a NumPy - nd array and split using 80-20 split, using the train_test_split function.

Task 2 : Eigenfaces Implementation.

Inbuilt “sklearn” library is used for the implementation of PCA. The Number of components chosen is taken to be 150 at first due to the following reasons.

Explained Variance Ratio: PCA helps us realize the amount of variance that is clarified by each primary component. One typical way to decide on the number of components is to select a sufficient number of components that can account for a notable part of the variability in the data. Here, I examined the explained variance ratio feature of PCA to make sure that a significant amount of the variance was preserved using 150 components.





Balance between Dimensionality Reduction and Information Retention: By selecting 150 components, our goal is to find a middle ground between decreasing the data's dimensionality and preserving sufficient information to accurately depict the faces in the dataset. This decision was made to avoid overfitting while capturing the key characteristics of the faces.

Computational Complexity: Adding more components can greatly raise computational complexity. By selecting 150 parts, we hope to find a good balance between how well the model works and how efficiently it runs.

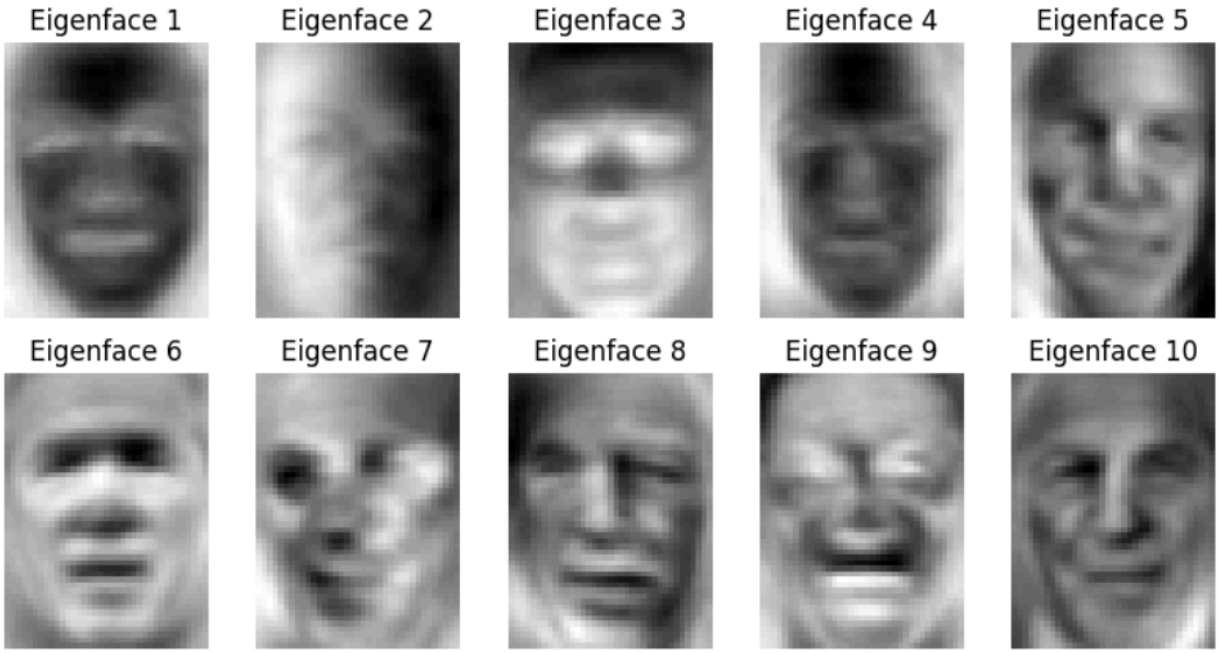
Task 3 : Model Training.

K-Nearest Neighbour Model has been chosen for training of the model using the transformed dataset.

Inbuilt KNN classifier in the “sklearn” library.

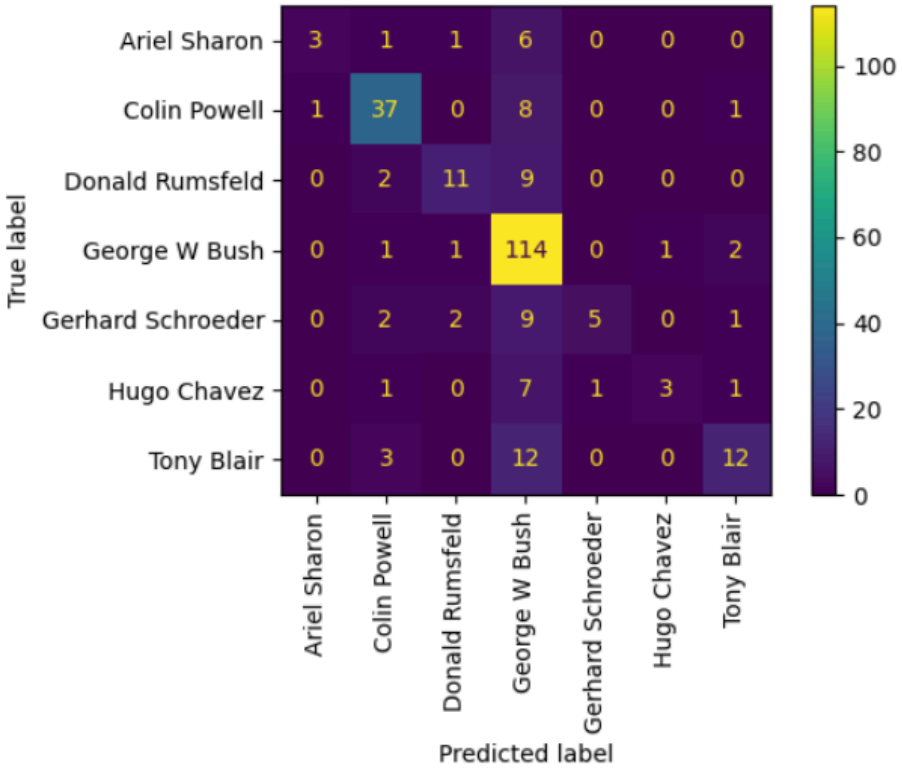
Task 4 : Model Evaluation.

The model Accuracy using KNN is coming out to be 71.70 percent.



Here are some of the eigen faces.

Here is the confusion matrix containing the labels of the faces involved.



The model performs really well on the face of George Bush. His face is rarely labeled as any other person's face.

The faces of Ariel Sharon are almost always classified incorrectly.

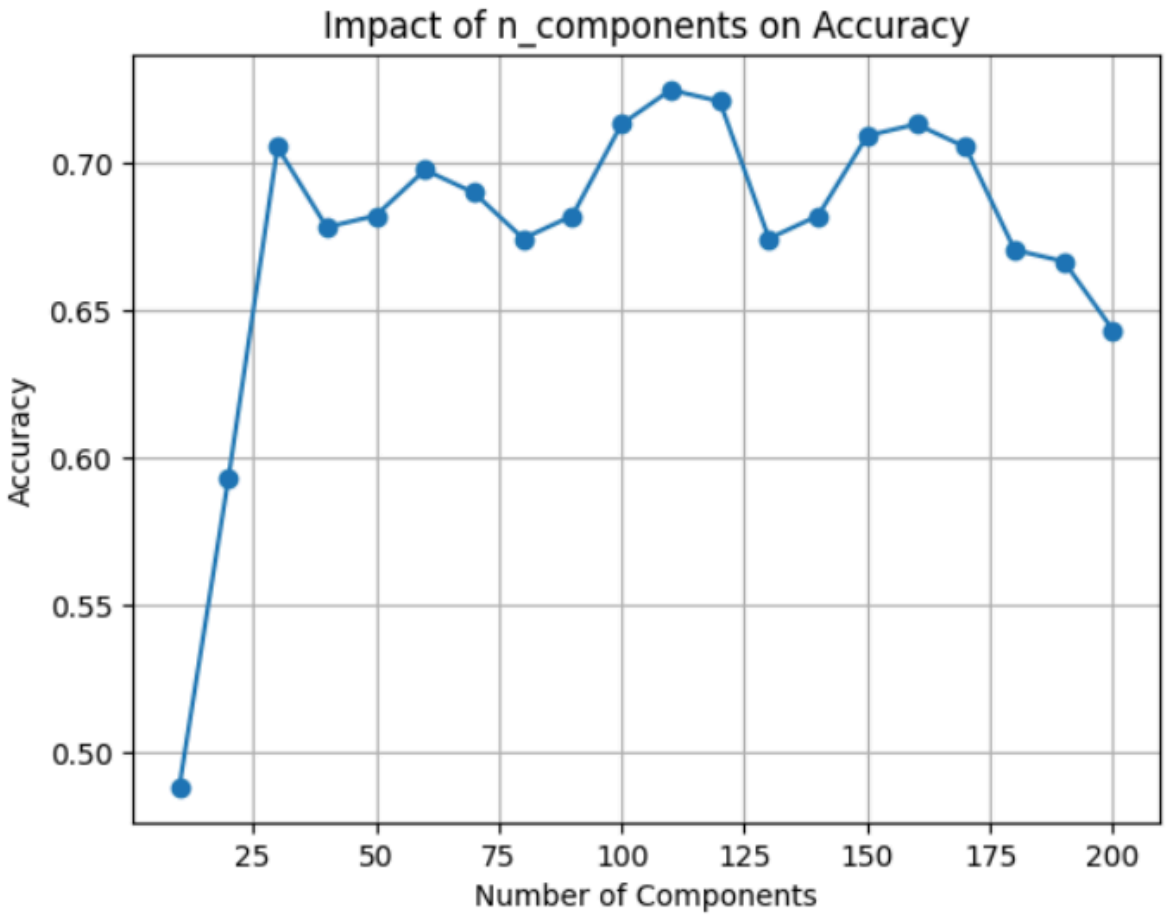
The f1-scores for Hugo Chavez and Ariel Sharon is the least, indicating the poor performance of the KNN model on these faces.

Ways to improve the model are as follows.

1. Increasing the training data: The number of faces with True Labels as Ariel Sharon and Hugo Chavex is very less in number. Hence the model is not able to train sufficiently for these faces.
2. Balancing the data: The dataset can be made more balanced, by including the same number of faces for all the labels.
3. Hyperparameter Adjustment: Try out various values of hyperparameters in the KNN classifier (such as number of neighbors, distance metric) using cross-validation to discover the best settings that enhance classification accuracy, particularly for Ariel Sharon and Hugo Chavez's faces.

Task 5 : Comparing using different values of n_components.

Here is the plot of accuracy vs n-components.



The accuracy fluctuates between 64% and 73%. The values almost remains constant for n_components >= 50