EE5026 Assignment: Face Recognition

QI SHUOLI | A0274285W

1. Introduction

1.1 Project Goal

For the course project, I constructed **a face recognition system**.

- I applied Principal Component Analysis (PCA) to perform data dimensionality reduction and visualization, to understand underlying data distribution.
- Then I train and apply three classification models, including Linear Discriminative Analysis (LDA), Support Vector Machine (SVM), and Convolutional Neural Networks (CNN), to classify the face images, and one clustering model – Gaussian Mixture Model (GMM) – to group the face images.

1.2 Programming Language

- Python

1.3 Dataset

- CMU PIE dataset
- Face photos taken by myself: I took 10 selfie photos for myself, converted to gray-scale images, and resized them into the same resolution as the CMU PIE images.

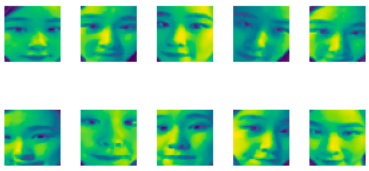


Figure 1: Face photos taken by myself (for EE6026 assignment using purposes only)

1.4 Data Pre-Processing: Split Training Data and Testing Data

- There are in total 68 different subjects in CMU PIE dataset. I chose 25 out of them randomly, which was collected using "random.seed(5026)" and that is [0, 4, 5, 8, 9, 11, 21, 22, 24, 27, 28, 33, 43, 44, 46, 47, 48, 49, 53, 55, 56, 57, 60, 65, 66, 68].

```
Subject [ 0  4  5  8  9 11 21 22 24 27 28 33 43 44 46 47 48 49 53 55 56 57 60 65
 66 68] are chosen.
```

Figure 2: Subject Choosing Results ("0" is the Face photos taken by myself)

- For each chosen subject, use 70% of the provided images for training and use the remaining 30% for testing.

2. PCA for Feature Extraction, Visualization & Classification

2.1 PCA Overview

- PCA is used to reduce the dimensionality of a collection of observations by finding a new set of variables smaller than the original set of variables. It could help capture big variability in data and ignore small variability.
- When using PCA, we compute the covariance matrix, perform eigenvalue decomposition, and output the PC matrix. It could find an orthonormal basis of data, sort dimension "importance", and discard low significant dimensions. It could get compact descriptions, ignore the noise, and hopefully improve the classification. However, it could not capture non-linear structure, and cannot get class labels.
- In this problem, the size of the raw face image is 32×32 pixels, resulting in a 1024-dimensional vector for each image. Randomly sample 500 images from the CMU PIE training set and my own photos. I first normalize the subset of images before applying PCA, to equalize feature scales.

2.2 Apply PCA to Reduce Dimensionality of Vectorized Images to 2

- Figure 3 shows those 500 images projected on 2-dimension space formed by the 2 first Principal Components. Images of the same subject are represented by the same marker. I highlighted the points corresponding to my photo as big black "+".

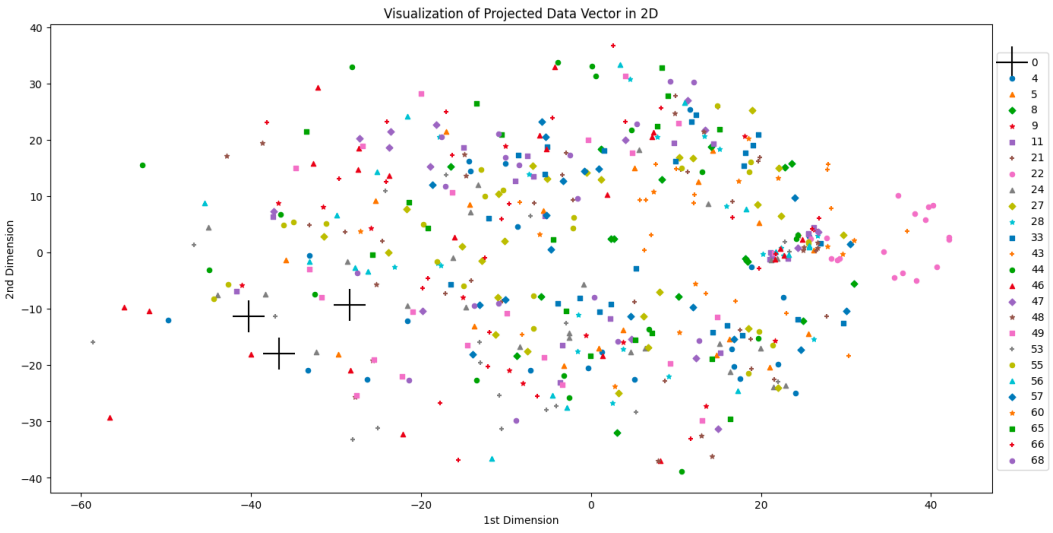


Figure 3: Apply PCA to reduce the dimensionality of vectorized images to 2.

2.3 Apply PCA to Reduce Dimensionality of Vectorized Images to 3

- Figure 4 shows those 500 images projected on 3-dimensional space formed by the 3 first Principal Components. Images of the same subject are represented by the same marker. I highlighted the points corresponding to my photo as big black "+".

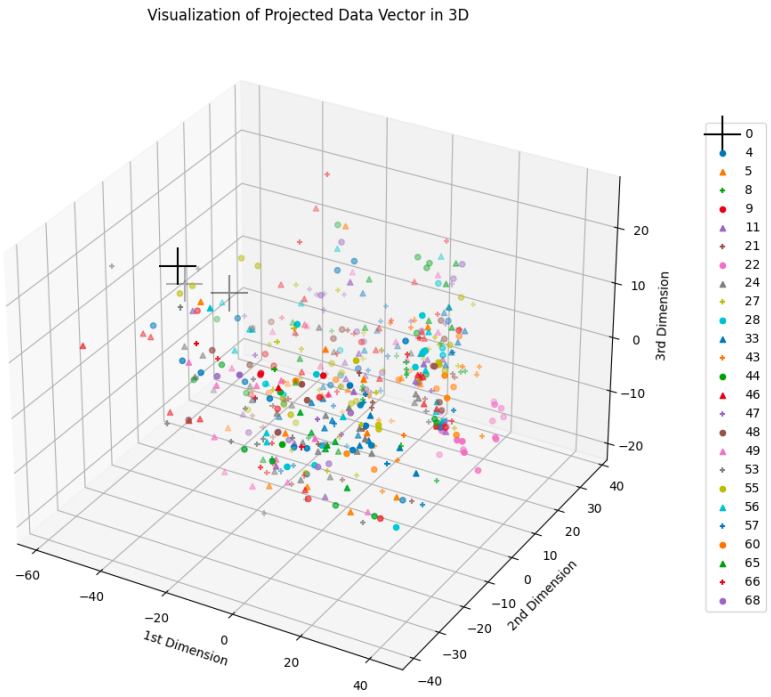


Figure 4: Apply PCA to reduce the dimensionality of vectorized images to 3

2.4 Visualize the Corresponding 3 Eigenfaces.

	Variance
PCA 0 Dimension	0.425
PCA 1 Dimension	0.260
PCA 2 Dimension	0.062

Table 1: Variance for each PC

Figure 5: Corresponding 3 eigenfaces

2.5 PCA + Nearest Neighbor Classification Results

- I applied PCA to reduce the dimensionality of face images to 40, 80, and 200 respectively. I classified the test images using the rule of nearest neighbor. The classification accuracy on the CMU PIE test images and my own photo is as follows.

	CMU PIE dataset	Own Photo
40 Dimension	0.826	0.
80 Dimensions	0.870	0.
200 Dimensions	0.888	0.

Table 2: Classification accuracy of PCA using the rule of the nearest neighbor.

3. LDA for Feature Extraction and Classification

3.1 LDA Overview

- LDA finds the most discriminative projection by maximizing between-class distance and minimizing within-class distance. LDA seeks to reduce dimensionality while preserving as much of the class-discriminatory information as possible.
- In this problem, I apply LDA to reduce data dimensionality from 2, 3, and 9.

3.2 Visualize Distribution with Dimensionality 2

- I visualize the distribution of the sampled data (as in the PCA section) with a dimensionality of 2 (like PCA). Figure 6 shows those 500 images projected on 2-dimensional space formed by the 2 first dimensions. Images of the same subject are represented by the same marker. I highlighted the points corresponding to my photo as big black "+".

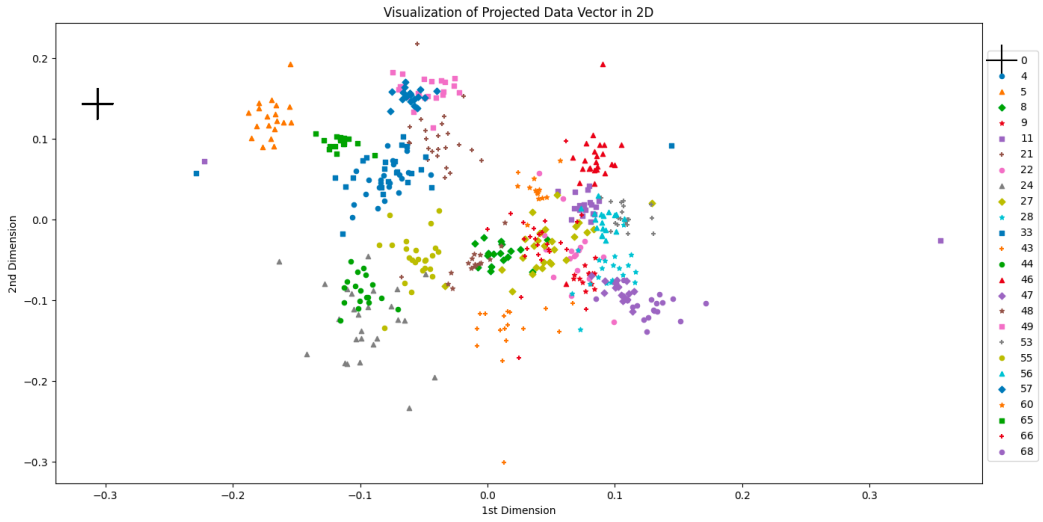


Figure 6: Visualize the distribution of the sampled data with dimensionality 2.

3.3 Visualize the Distribution with Dimensionality 3

- I visualize the distribution of the sampled data (as in the PCA section) with a dimensionality of 3 (like PCA). Figure 7 shows those 500 images projected on 3-dimensional space formed by the 3 first dimensions. Images of the same subject are represented by the same marker. I highlighted the points corresponding to my photo as big black "+".

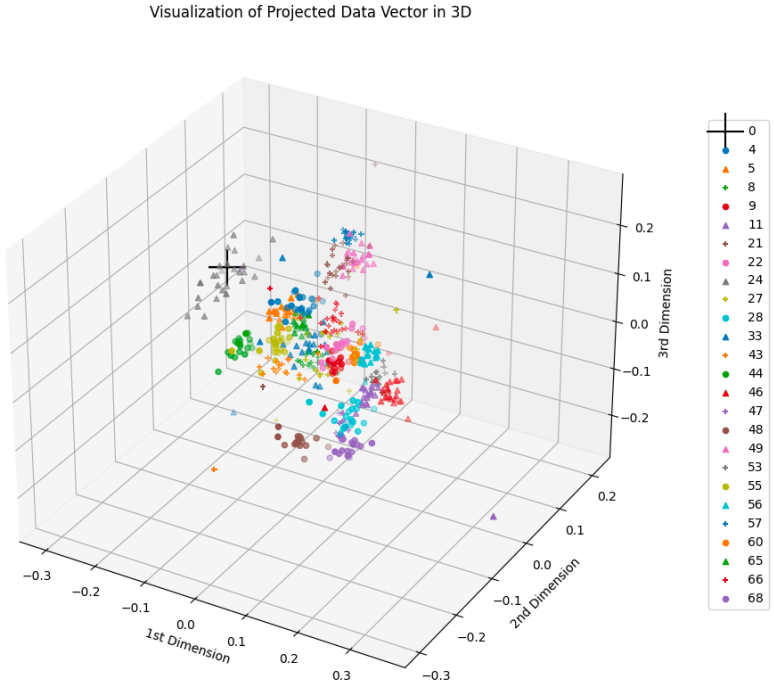


Figure 7: Visualize the distribution of the sampled data with dimensionality 3

3.4 Apply LDA to Reduce Data Dimensionality from 2, 3, and 9

- Report the classification accuracy for data with dimensions of 2, 3, and 9 respectively, based on nearest neighbor classifier. Report the classification accuracy on the CMU PIE test images and your own photo separately.
- We can also observe that the accuracy increases when the number of LDA features increases. However, the performance for own photo is 0., which matches the conclusion from the essay "A. Martinez, A. Kak, "PCA versus LDA", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 2, pp. 228-233, 2001.", that is, when the training set is small, PCA can outperform LDA.

	CMU PIE dataset	**Own Photo**
2 Dimensions	0.453	0.
3 Dimensions	0.626	0.

9 Dimensions	0.915	0.

Table 3: Classification accuracy of LDA using the rule of nearest neighbor.

4. SVM for Classification

4.1 SVM Overview

- A Support Vector Machine is a powerful supervised machine learning algorithm used for classification and regression tasks. Its primary objective is to find an optimal hyperplane in a high-dimensional space that best separates different classes of data.

4.2 SVM Result

- I use the raw face images (vectorized) and the face vectors after PCA pre-processing (with dimensionality of 80 and 200) as inputs to linear SVM. Try values of the penalty parameter C in $\{ 1 \times 10^{-2}, 1 \times 10^{-1}, 1 \}$. Here is the classification accuracy with different parameters and dimensions.
- Discuss the effect of data dimension and parameter C on the final classification accuracy: According to the results, in this specific case, varying C does not have a substantial impact on classification accuracy. Across different values of C, the accuracy remains consistently high (around 98%). To be specific, PCA (D-80/200) is already reaching a very high accuracy when $C = 1 \times 10^{-1}$. This suggests that the SVM model is robust to changes in the regularization parameter within the specified range. The effect of data dimensionality reduction is also evident. As expected, PCA (D-80) and PCA (D-200) maintain high classification accuracies, demonstrating that the reduced-dimensional representations still capture the essential information for effective face recognition. The marginal drop in accuracy from raw face images to PCA-processed data.
- The result indicates that the data is well-behaved, and the model is not prone to overfitting even with larger C values. Additionally, the robust performance across different dimensions (raw face images, PCA (D-80), PCA (D-200)) suggests that the SVM model, combined with PCA dimensionality reduction, is effective in handling variations in input data while maintaining high accuracy.

	Raw Face Images	**PCA (D-80)**	**PCA (D-200)**
$C = 1 \times 10^{-2}$	0.985	0.975	0.980
$C = 1 \times 10^{-1}$	0.985	0.977	0.984
$C = 1$	0.985	0.977	0.984

Table 4: Classification accuracy of SVM

5. Neural Networks for Classification

5.1 CNN Overview

- I trained a CNN with two convolutional layers and one fully connected layer, with the architecture specified as follows: number of nodes: 20-50-500-26. The number of the nodes in the last layer is fixed as 26 as we are performing a 26-category (25 CMU PIE faces plus 1 for yourself) classification. Convolutional kernel sizes are set as 5. Each convolutional layer is followed by a max pooling layer with a kernel size of 2 and a stride of 2. The fully connected layer is followed by ReLU. The final classification performance for testing data is 0.990.

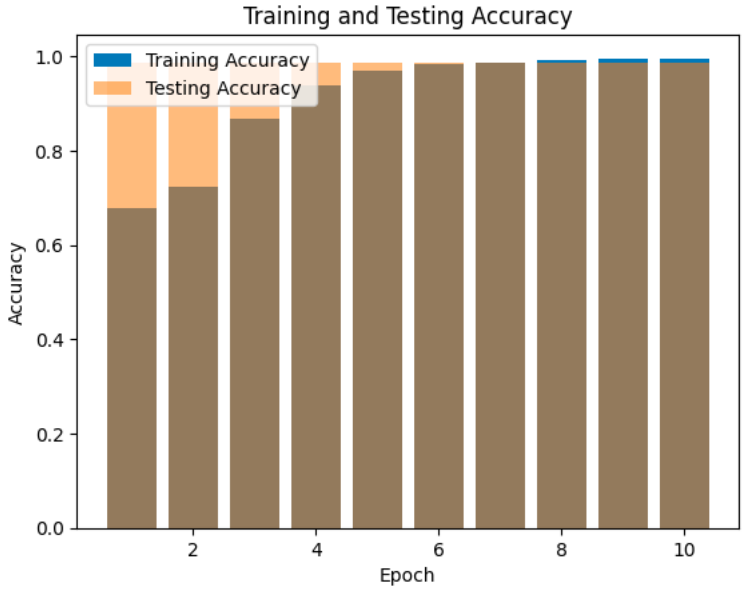


Figure 11: Training and testing accuracy.

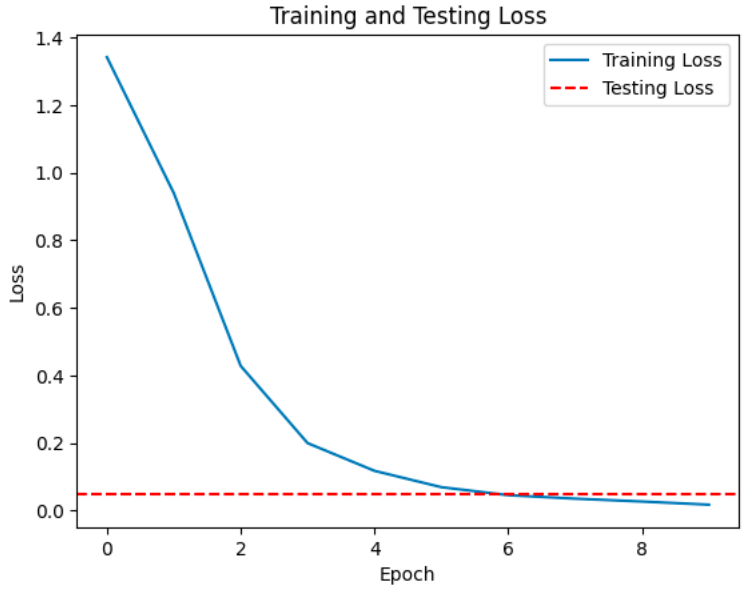


Figure 12: Training and testing loss.

6. Appendix:

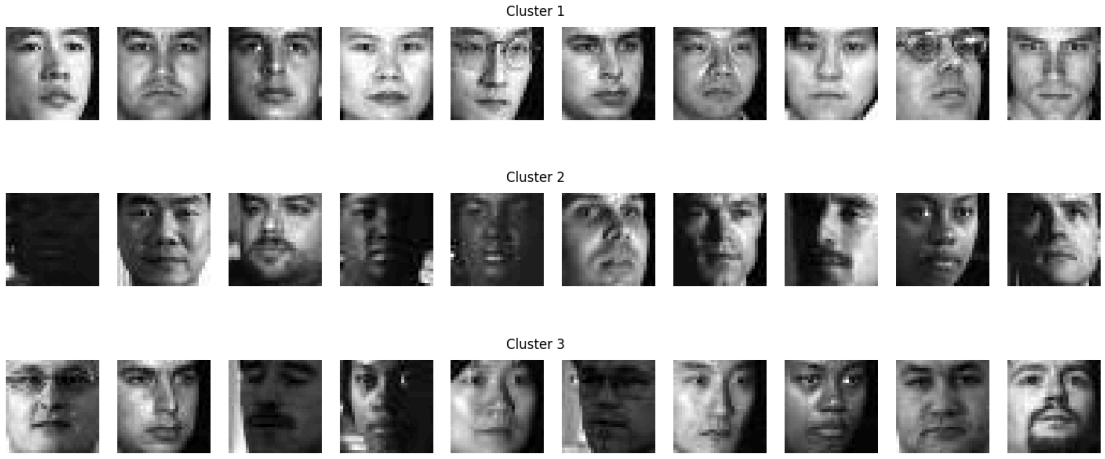
As it says, “In your experiments, PCA and LDA are mandatory. As for GMM, SVM and CNN, choose 2 of them based on your interest.”, I performed SVM and CNN for this project, but actually I also did GMM additionally out of interests, and here is the results.

6.1 GMM Overview

- GMM is to measure the weighted sum of a number of Gaussians where the weights are determined by a distribution. Rather than identifying clusters by “nearest” centroids, it fit a Set of Gaussians to the unlabeled data and get Maximum Likelihood over a mixture model.

6.2 Raw Data + GMM

- I use the raw face images as the provided training data. I trained a GMM model with 3 Gaussian components on these data. Then, I visualized the clustering results from GMM, i.e., visualize the images that are assigned to the same component.



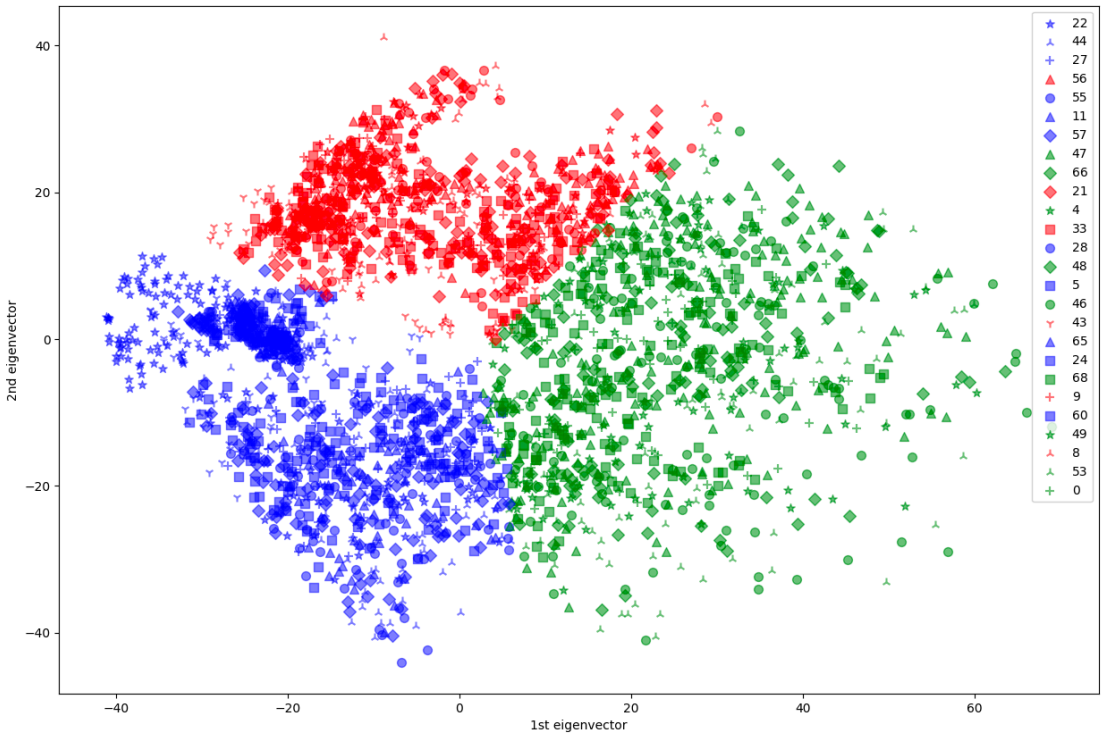


Figure 8: GMM for raw data.

6.3 PCA D-80 + GMM

- I use the face vectors after PCA pre-processing with a dimensionality of 80 as the provided training data. I trained a GMM model with 3 Gaussian components on these data. Then, I visualized the clustering results from GMM, i.e., visualize the images that are assigned to the same component.

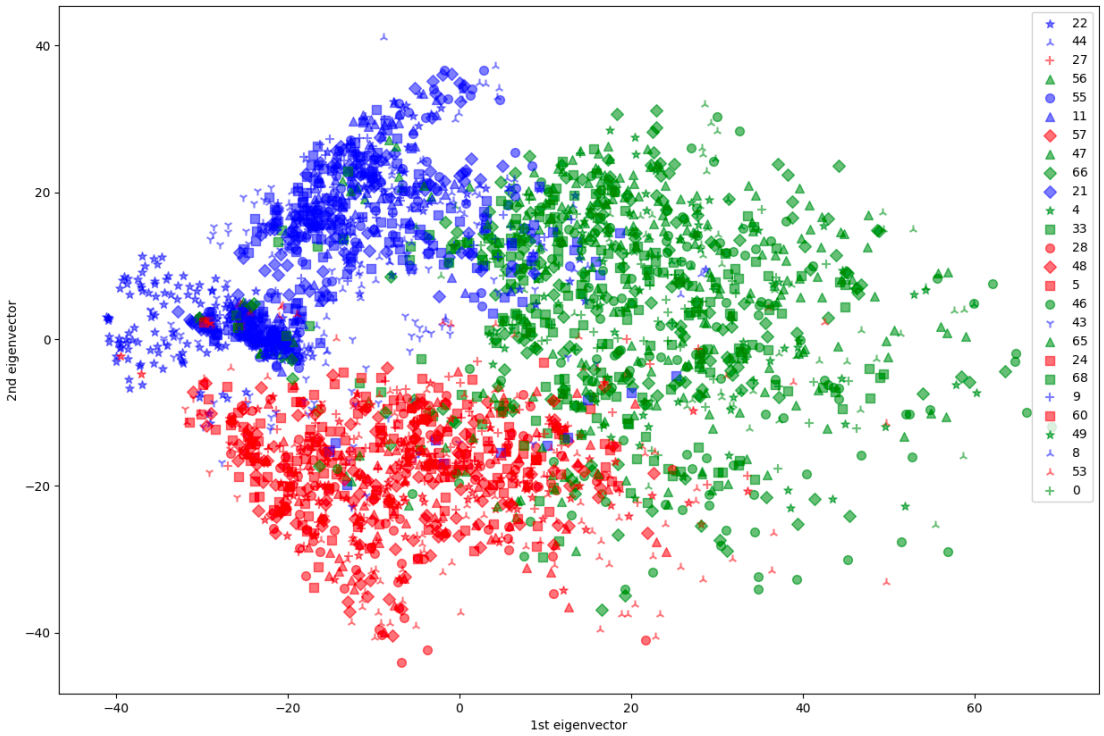


Figure 9: GMM after PCA pre-processing with dimensionality of 80.

6.4 PCA D-200 + GMM

- I use the face vectors after PCA pre-processing with a dimensionality of 200 as the provided training data. I trained a GMM model with 3 Gaussian components on these data. Then, I visualized the clustering results from GMM, i.e., visualize the images that are assigned to the same component.

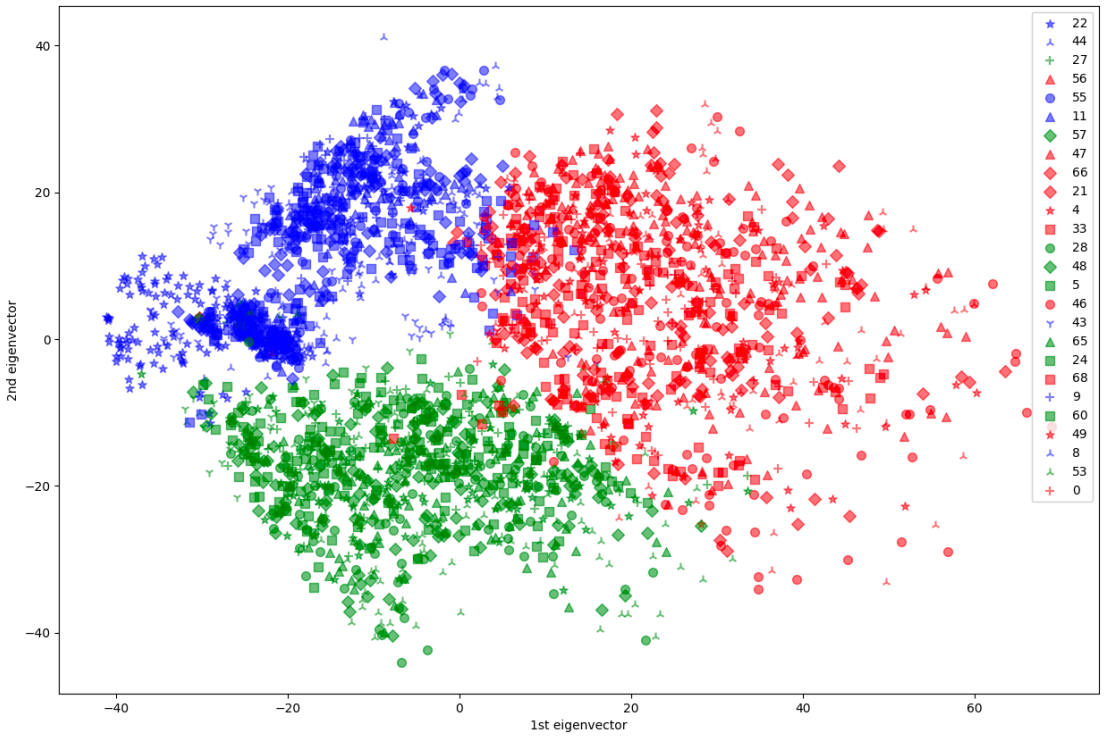


Figure 10: GMM after PCA pre-processing with dimensionality of 200.