Report of CA2 EE5907 Pattern Recognition

Li Ruoyu

November 13, 2023

1 Introduction

Goal For this course project, students will work individually to construct a face recognition system. Students are required to apply Principal Component Analysis (PCA) to perform data dimensionality reduction and visualization, in order to understand underlying data distribution. Then students are required to train and apply three classification models – Linear Discriminative Analysis (LDA), Support Vector Machine (SVM) and Convolutional Neural Networks (CNN) – to classify the face images, and one clustering model. Through the project, students are expected to gain basic and important knowledge of currently popular pattern recognition techniques.
Programming Language Students may use any language of their choosing for the project, though at least starting with the MATLAB or Python. The main language used in this report is python

2 Dataset

The project will be conducted on the CMU PIE dataset and the face photos taken by myselves. There are in total 68 different subjects. I choose 25 out of them by random and indicate my choice in the report. For each chosen subject, use 70% of the provided images for training and use the remaining 30% for testing. Besides the provided CMU PIE images, I take 10 selfie photos for myself, convert to gray-scale images, resize them into the same resolution as the CMU PIE images and split them into 7 for training and 3 for testing. Put the 7 selfie-photos into the training set and 3 into the test set. My selfie-photo dataset are show in Figure 1.

3 Results

3.1 PCA

The size of 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. Apply PCA to reduce the dimensionality of vectorized images to 2 and 3 respectively. Visualize the projected data vector in 2d and 3d plots. Highlight the projected points corresponding to my photo. Also visualize the corresponding 3

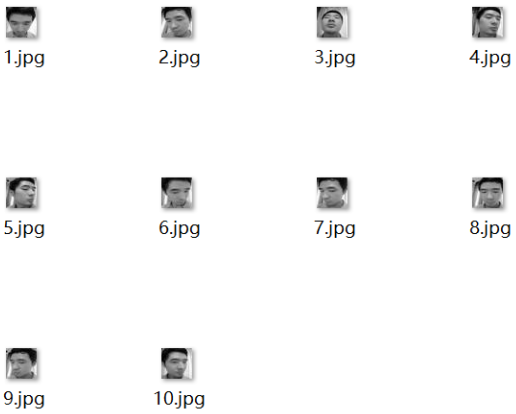

Figure 1: Selfie-photos.

eigenfaces used for the dimensionality reduction.The folders randomly selected in the PCA experiment were shown in Figure 2.

```
import random

random_numbers = random.sample(range(1, 69), 25)
print(random_numbers)
```

```
[1]
[30, 26, 28, 42, 17, 6, 66, 59, 25, 12, 19, 36, 64, 53, 5, 32, 67, 29, 2, 61, 35, 48, 43, 37, 22]
```

Figure 2: PCA dataset.

3.1.1 PCA dimensionality reduction and visualization

The visualisation results are shown in Figure 3. The corresponding 3 eigenfaces used for the dimensionality reduction are shown in Figure 4.

3.1.2 PCA + KNN

Then apply PCA to reduce the dimensionality of face images to 40, 80 and 200 respectively. Classifying the test images using the rule of nearest neighbor. Show the classification accuracy on the CMU PIE test images and my own photo seperately. The results shown in the Figure 5 use the KNN algorithm K to be 1.

3.2 LDA

Apply LDA to reduce data dimensionality from to 2, 3 and 9. As a precautionary note, according to the title, I only selected 20 images from each folder in the PCA, and subsequent experiments will use all the images in the selected folders. Visualize distribution of the sampled data (as in the PCA section) with dimensionality of 2 and 3 respectively (similar to PCA). Report the classification accuracy for data with dimensions of 2, 3 and 9 respectively, based on nearest neighbor classifier. Show the

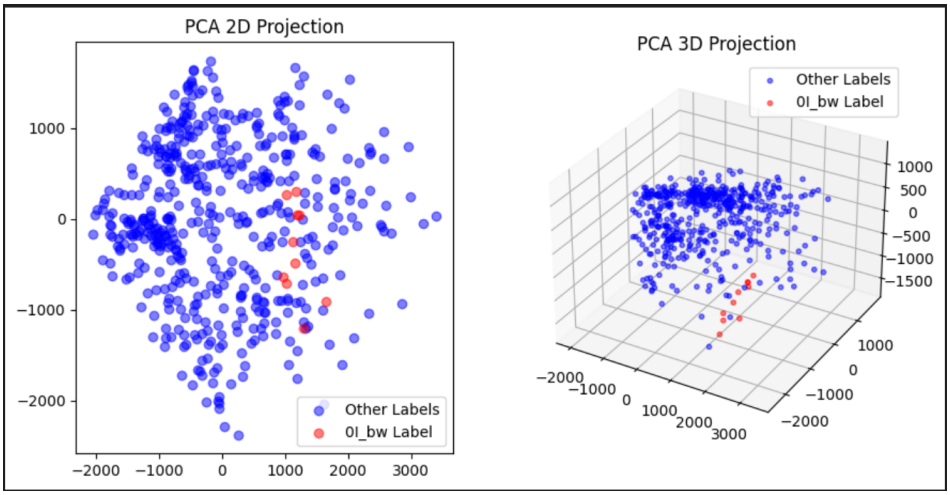

Figure 3: PCA based data distribution visualization.

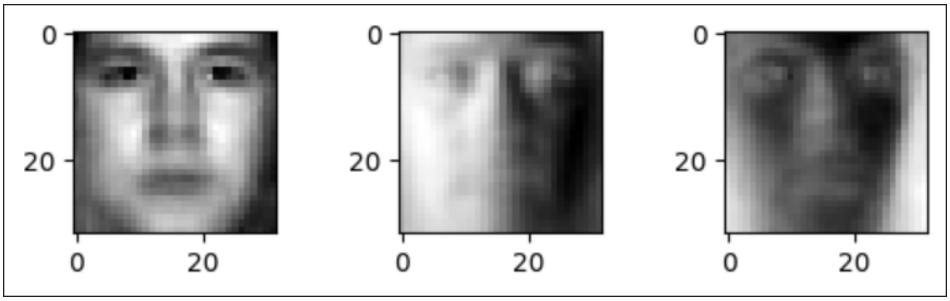

Figure 4: The corresponding 3 eigenfaces used for the dimensionality reduction

classification accuracy on the CMU PIE test images and my own photo seperately. The folders randomly selected in the LDA experiment were shown in Figure 6.

3.2.1 LDA dimensionality reduction and visualization

The visualisation results are shown in Figure 7.

3.2.2 LDA + KNN

The accuracy results shown in the Figure 8 use the KNN algorithm K to be 1. It can be clearly seen that compared with PCA, LDA can achieve higher accuracy in lower dimensions.

3.3 SVM

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 $\{ 1\times10^{-2}, 1\times10^{-1}, 1 \}$. Report the classification accuracy with different parameters and dimensions. Discuss the effect of data dimension and parameter C on the final classification accuracy. The folders randomly selected in the SVM experiment were shown in Figure 9.

```
accuracy: 0.425, Myaccuracy: 1.000 in the case of dimension 40
accuracy: 0.464, Myaccuracy: 1.000 in the case of dimension 80
accuracy: 0.484, Myaccuracy: 1.000 in the case of dimension 200
```

Figure 5: Accuracy of PCA+KNN

```
import random

random_numbers = random.sample(range(1, 69), 25)
print(random_numbers)

[2]
...  [33, 30, 25, 37, 4, 66, 43, 35, 48, 47, 46, 42, 19, 12, 60, 53, 55, 26, 49, 14, 58, 15, 11, 36, 3]
```

Figure 6: LDA dataset.

As far as the experimental results shown in Figure 10 are concerned, I think that the parameter C has no effect on the classification accuracy, and a fairly high accuracy can be achieved when the dimension is 80, and continuing to improve the size of the dimension may be able to further improve the accuracy, but there have been cases in which the accuracy has decreased in the course of the experiments.

3.4 CNN

Train 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 26-category (25 CMU PIE faces plus 1 for yourself) classification. The folders randomly selected in the CNN experiment were shown in Figure 11. Convolutional kernel sizes are set as 5. Each convolutional layer is followed by a max pooling layer with a kernel size of 2 and stride of 2. The fully connected layer is followed by ReLU. Train the network and report the final classification performance. The results are shown in the Figure 12.

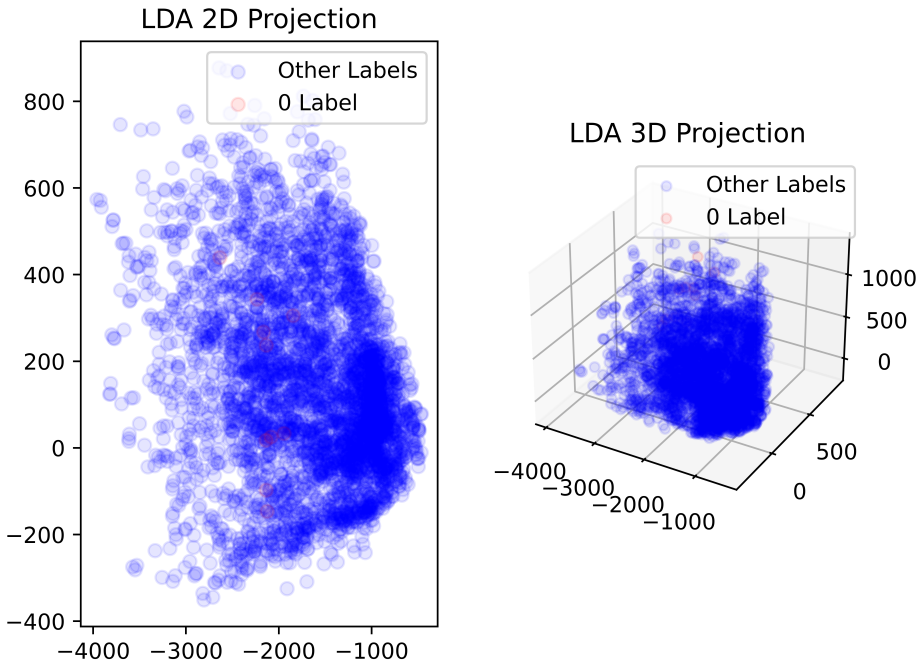

Figure 7: LDA based data distribution visualization.

```
accuracy: 0.156, Myaccuracy: 0.000 in the case of dimension 2
accuracy: 0.377, Myaccuracy: 0.667 in the case of dimension 3
accuracy: 0.863, Myaccuracy: 1.000 in the case of dimension 9
```

Figure 8: Accuracy of LDA+KNN

```
import random

random_numbers = random.sample(range(1, 69), 25)
print(random_numbers)
```

```
[1]
...  [6, 20, 22, 16, 63, 52, 48, 53, 1, 18, 9, 17, 41, 24, 35, 40, 45, 36, 30, 15, 47, 2, 66, 10, 7]
```

Figure 9: SVM dataset

```
...   C =  0.01
      Accuracy = 99.061% (1266/1278) (classification)
      Myaccuracy: 0.667 in the case of dimension 80
      C =  0.1
      Accuracy = 99.061% (1266/1278) (classification)
      Myaccuracy: 0.667 in the case of dimension 80
      C =  1
      Accuracy = 99.061% (1266/1278) (classification)
      Myaccuracy: 0.667 in the case of dimension 80
      C =  0.01
      Accuracy = 99.1393% (1267/1278) (classification)
      Myaccuracy: 0.667 in the case of dimension 200
      C =  0.1
      Accuracy = 99.1393% (1267/1278) (classification)
      Myaccuracy: 0.667 in the case of dimension 200
      C =  1
      Accuracy = 99.1393% (1267/1278) (classification)
      Myaccuracy: 0.667 in the case of dimension 200
```

Figure 10: Accuracy of SVM

```
random_numbers = [6, 20, 22, 16, 63, 52, 48, 53, 1, 18, 9, 17, 41, 24, 35, 40, 45, 36, 30, 15, 47, 2, 66, 10, 7]
```

Figure 11: CNN dataset

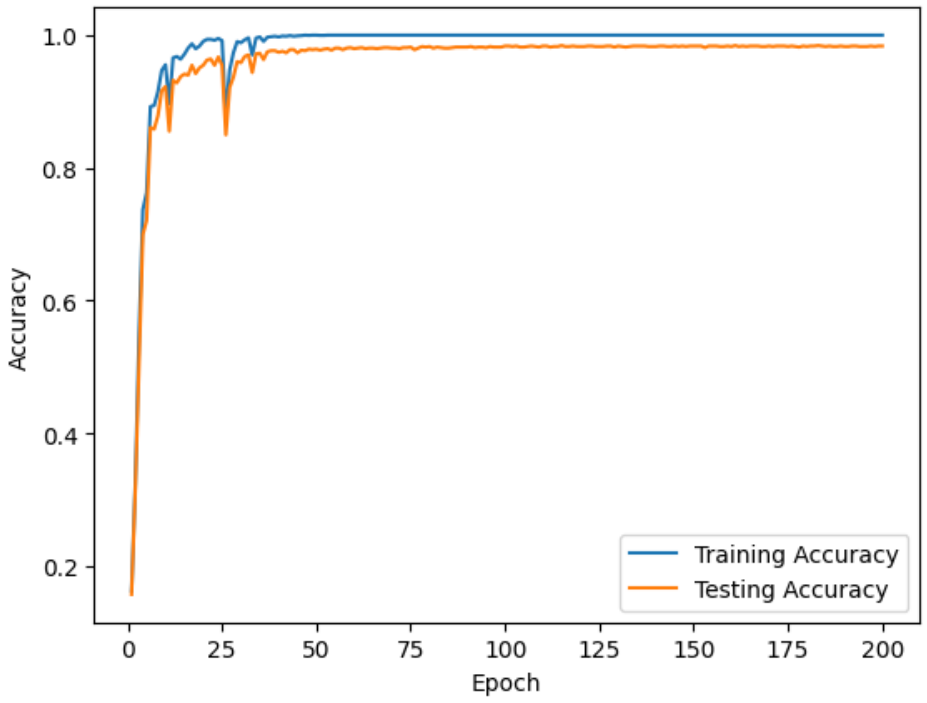

Figure 12: Accuracy of CNN