EE5907/EE5027 Pattern Recognition Programming Assignment CA2 Report

Shuo SUN A0162488U
email shuo.sun@u.nus.edu

4 November 2021

Abstract

This report documents the experiment results and observations from EE5907/EE5027 Pattern Recognition Programming Assignment 2, Face Recognition. The aim of the project is to apply different methods to classify the classes from face images. In this project, Five popular algorithms that are used in (image) data classification and clustering tasks, namely: Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA), Gaussian Mixture Model (GMM), Support Vector Machine (SVM), and Convolutional Neural Network (CNN), were implemented and evaluate on the same Face Dataset. The performances of the above methods and key observations made during experiment are presented and discussed in this report. The original project code has been open-sourced and can be found on GitHub https://github.com/SS47816/face-recognition

Dataset Overview

The first part of the dataset used in this programming assignment was The CMU Pose, Illumination, and Expression (PIE) Database [1], which contains more than 40,000 facial images of 68 people(classes). Within each class, there are 170 images of the person taken from different angles, with different expression and under various illuminations, posing challenges for classification tasks. All the images were converted to gray-scale and cropped to the size of 32x32. The second part of the dataset used in this programming assignment was created using selfies, which imitate the styles and formats used in the CMU PIE Dataset. Subsequently these images were combined with the CMU PIE dataset as a individual class.

Among the 68 classes in the CMU PIE dataset, 25 classes were randomly selected and combined with the customized dataset for this project. Within the combined dataset, for each class, 70% of the images were randomly selected to form the training set while the rest 30% of the images were used as the test set. Throughout the experiment, the above mentioned training and test sets with 25+1 classes were used by the first four algorithms, PCA, LDA, GMM, and SVM, while the last algorithm CNN used a slightly smaller dataset with only 20+1 classes.

1 PCA for Feature Extraction, Visualization and Classification

Principal Component Analysis (PCA) is a widely-used dimentionality reduction technique in data processing. It can extract the most significant components from high dimensional data, which made it a popular data pre-processing technique and compression technique in a wide range of applications.

1.1 Implementation & Experiment

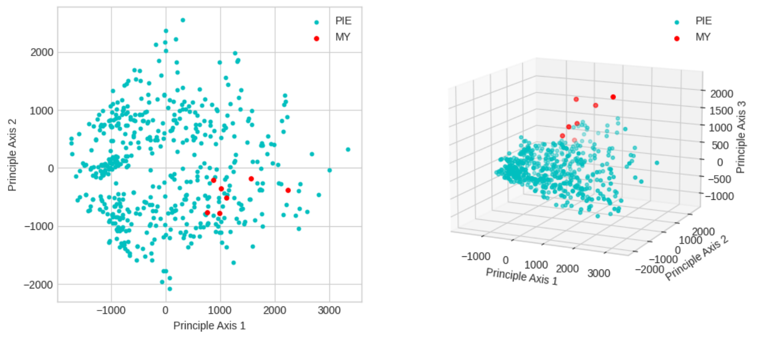

Figure 1: PCA projected data vectors in 2D and 3D

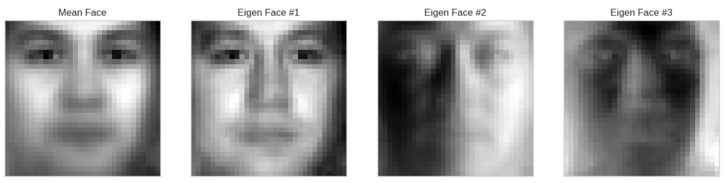

Figure 2: PCA mean and eigen faces visualization

In this experiment, all the input data, which were gray-scale images of the size $32 \times 32 \times 1$, were first stretched into a 1D vector of the size 1024. After pre-processing, a PCA was applied to these image vectors. The first (two)three most significant components were selected as the (two)three axis of the plot and projected the image vectors onto these axis, as shown in Figure 1. Notably that only 500 images from the CMU PIE training set were used to fit the PCA model, and later the PCA model was tested on the whole test set which had 1298 test images.

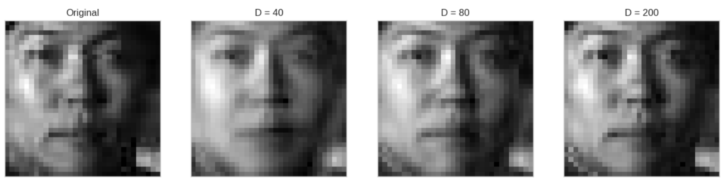

Figure 3: PCA reconstructed faces

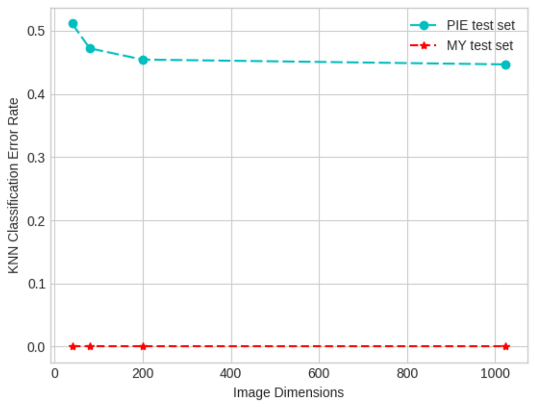

Figure 4: PCA KNN classification results

At the same time, the "Mean Face", which is a virtual face image composed by the mean values of all the training image data, and 3 "Eigen Faces", which are the 3 most significant components extracted from the image vectors by PCA, are computed and visualized in Figure 2.

Next, PCA was applied to reduce the dimentionalities of the input images from 1024 to 40, 80 and 200 respectively, and feed into a K-Nearest Neighbours Classifier to perform classification task. During this step, for each image vector went through the PCA process, their top 40, 80, or 200 components were used respectively in each classification experiment. Meanwhile, by applying a inverse projection on these compressed image vectors, faces were able to be reconstructed as the example showed in Figure 3. The error rates of the KNN classification on each batch of PCA dimentionality-reduced data were collected and plotted in Figure 4. In addition, a baseline KNN classification test was conducted on the raw image vectors of the size 1024, without any PCA processing, and added to Figure 4 for comparison.

1.2 Observations & Discussion

From Figure 1, it can be observed that in the 2D plot on the left, the red data points denoting the selfie photos blended into the CMU PIE dataset very well, making it difficult to draw a clear boundary between the two groups, which also proved that the carefully created selfie photos dataset resembled the original CMU PIE dataset quite well. As far as the first two principle components can tell, the difference between the two groups is very small. By looking at the 3D plot in Figure 1, it was noticed that on the third principle component, the two groups of data started to separate, with selfie photos having higher values on the z-axis. From the KNN algorithm's perspective, all the data points belonging to the selfie class are scattered not far from each other, providing convenience for the KNN algorithm to produce correct classifications.

As shown in Figure 2, these four faces (1 mean face, and 3 eigen faces) all represented important features extracted based on the entire dataset. They can be viewed as statistical summaries of all the faces in the dataset in some way, with each emphasising on one important aspect. For example, the most important eigen face is very similar to the mean face, which both showed all the important common features on a human face such as eyes, noses, mouths, outline, etc. While the second and third eigen faces emphasised more on the change in view perspective/ angle of illumination, which are key features in the CMU PIE dataset. This showed the PCA's strong capabilities in extracting the most important information/features from a mass amount of data with high dimensions.

Figure 3 showed a comparison between the original face image and the reconstructed face images. It can be clearly observed that, as more and more eigen faces overlaid onto the reconstructed image, the face is gradually becoming more distinguishable with more details and closer to the original image. Even with a very low number of dimensions compared to the original image (1024), the reconstructed face images were able to restore most of the details on the original image and made the face identifiable for human eyes. The result proved that PCA can be a excellent choice for data dimensionality reduction for its ability to preserve most important features in data while achieving a high reduction in data dimensionalities.

As showed in Figure 4, the KNN classifier achieve an error rate of 0.0 on the selfie class throughout the experiment with any number of dimensions, suggested that this class might be too easy for the KNN algorithm. This high performance on the the selfie photos is highly predictable since the total number of train and test photos is merely 10, and in Figure 1, PCA results already showed that the selfie photos can be clearly separated from the rest of the classes. The KNN's prediction error rate on other CMU PIE classes could be statistically more reliable and served as a better performance indicator. On the CMU PIE dataset, the KNN's error rate curve decreased steeply from 0.512 to 0.472 at first, then gradually tended to be flat, approximating a error rate of 0.447 at the end. As expected, with a larger number of data components, the classification error tended to be smaller, gradually approximating the baseline error rate which was taken on the original image. This trend showed that compared the principle components (or eigen faces) at the tail part, the top principle components contributed more on the classification accuracy and contained more important information about the face image. In practice, a similar curve plot could be used as a reference to help find a balance between the number of data dimensions and the accuracy required by the application. In general, the results were not satisfactory partially due to the fact that only a small training set (with merely 500 randomly selected images) were used to fit the PCA model, which could not generalize well on th test set (with 1298 images).

2 LDA for Feature Extraction and Classification

Linear Discriminant Analysis (LDA) is a classification technique in data processing. Different from the PCA method, which tries to find the axis with the largest covariance within a group of data points, LDA aims to find the axis that create the largest covariance between data groups after a projection. To maximize the separation between data groups, LDA uses a scatter matrix that represents "between class" and "within class" scatter. Potentially, LDA can be more efficient and specialised in classification tasks comparing to PCA method.

2.1 Implementation & Experiment

In this round of experiment, all the input data went through the same vectorization process as described in the previous section before feeding into the LDA model. Similarly, the first (two)three most significant axis were selected as the (two)three axis of the plot and projected the image vectors onto these axis, as shown in Figure 5.

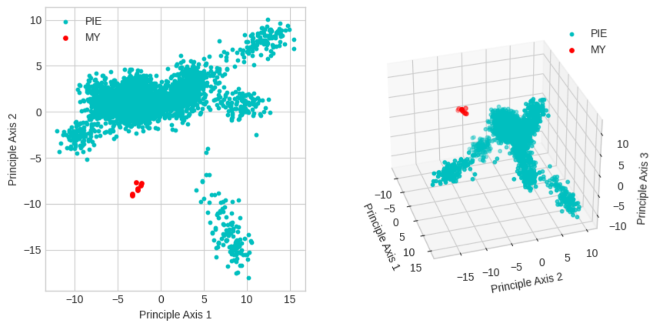

Figure 5: LDA projected data vectors in 2D and 3D

Next, dimentionality reduction was performed on the data using LDA. The input image vectors with 1024 dimensions were reduced to to 2, 3 and 9 respectively, and feed into the same K-Nearest Neighbours Classifier as in the previous section to perform classification task. The KNN classifier was first fit with the dimentionality-reduced training set (2946 images), then tested on the test set (1298 images). Results were collected and plotted in Figure 6.

2.2 Observations & Discussion

From Figure 5, it can be observed that the separation between different groups of data were quite significant on the plot, especially along the axis 2 direction. Similar to the plot in the

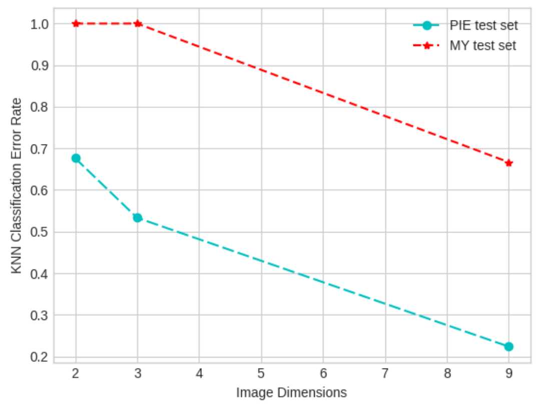

Figure 6: LDA KNN classification results

previous section, the selfie data points scattered very close to each other, even closer than in the previous method. Other CMU PIE classes also tended to form several clusters on the plot.

In Figure 6, the KNN classifier performed poorly on the selfie class, with a error rate of 1.0 at fisrt, then reduced to 0.667 at $D = 9$. However, the other line representing the classification error rate on the PIE dataset performed quite well and showed that the the error rate decreased quickly as the number of data dimensions increases. Noticeably that the classification error rates on the CMU PIE dataset this time at $D = 2$ and $D = 3$ were 0.676 and 0.533 respectively, which were quite inaccurate. However, LDA was able to drop this error rate quickly to 0.223 when the dimentionality came to $D = 9$, outperformed the previously method PCA-KNN which used more dimensions. This resulted showed that in terms of classification tasks, LDA might be able to achieve higher accuracy with even less data dimensions use than that in PCA. Following the trend on the plot, potentially the error rate could be further improved by a certain margin if the number of dimensions could continue growing for a certain range.

3 GMM for Clustering

3.1 Implementation & Experiment

Gaussian Mixture Models (GMM) is one of the clustering algorithms that clusters data based on a probabilistic approach. GMM assumes a finite known number of clusters exists in the given dataset and are distributed in a manner that assembles Gaussian distribution. During the training, the GMM is able to learn the parameters that define these Gaussian distribution and compute the probability of a data point belonging to a certain cluster.

Before conducting the GMM clustering, all the input data went through the same vectorization process and a PCA dimensionality reduction pre-processing technique, forming image vectors with dimensionality of 80 and 200 respectively. A GMM with 3 Gaussian components were train on the given training set data. Figure 7 and Figure 8 showed the sample images randomly selected from each cluster on the D=80 and D=200 training sets respectively.

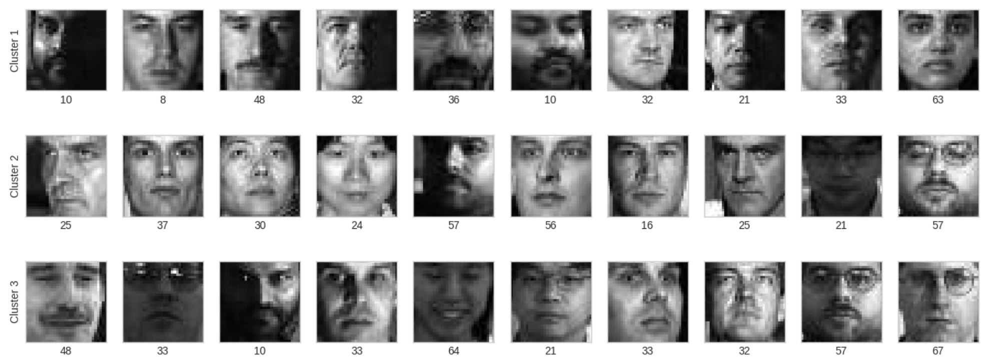

Figure 7: GMM clustering results (D=80)

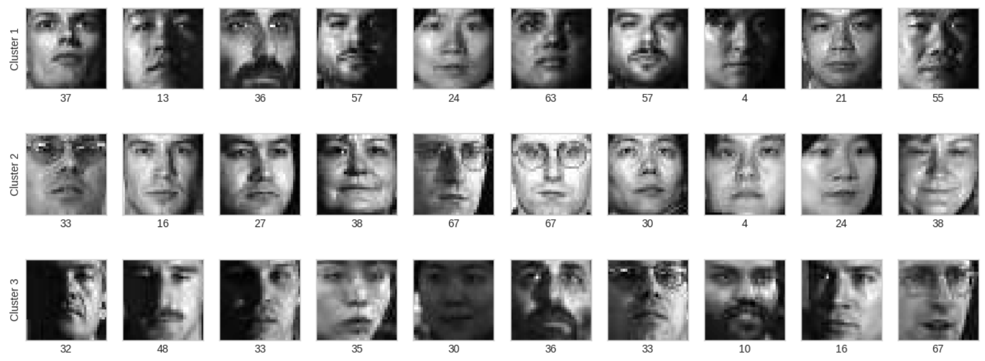

Figure 8: GMM clustering results (D=200)

3.2 Observations & Discussion

In Figure 7 and Figure 8, the number below each image were the class labels of that image. Based on the results, it can be observed that, most of the images with the same labels were clustered into the same cluster, for example, the three images with the label 33 were clustered into "Cluster 3" in Figure 7. However, there are also exceptions where images from the same class were clustered into different clusters, since the number of clusters used in this experiment was only 3. However, despite the inaccuracy and ambiguity in the way that these faces were clusters, certain trends are still revealed by these images. For example, there are a certain degrees of similarities in the face pose angles and the angles illumination within each cluster. In general, GMM can be a excellent choice of clustering method if the distribution of the data follows or is close to Gaussian distribution. It can also be used as a data pre-processing technique. However, one of the drawbacks is that the number of clusters needed to be determined before applying the method.

4 SVN for Classification

Support Vector Machine (SVM) is one form of supervised machine learning algorithm for classification and regression tasks. It aims to find the best hyper-plane that divides different groups of data by maximizing the distances from the nearest data points (support vectors). SVM is popular with a wide range of data classification tasks for its ability to deal with complex data. In practice, it is often used together with all forms of kernels which can help transform the data into higher dimensions, making the classification more accurate.

4.1 Implementation & Experiment

Table 1: SVM classification error rates

	D=80		D=200	
C Value	PIE Testset	MY Testset	PIE Testset	MY Testset
0.01	0.961	0.000	0.961	0.000
0.10	0.476	0.000	0.465	0.000
1.00	0.067	0.000	0.056	0.000

Similar to the previous experiment, all the input data went through the same vectorization process and a PCA dimensionality reduction pre-processing technique, forming image vectors with dimensionality of 80 and 200 respectively. Different SVMs with a penalty parameter C value of 0.01, 0.1, and 1.0 were tested in this experiment on the two pre-processed datasets with different dimensions. The classification error rates recorded were showed in Table 1.

4.2 Observations & Discussion

Based on the data showed in Table 1, it can be observed that the value of penalty parameter C affected the classification significantly. Since all the SVM classifier reached 0.0 error rate on the selfie photos, the results on the CMU PIE dataset could be a better indicator of performance of the classifiers. By comparing the difference in error rates between different C values along each column, it can be observed that within the range between 0.01 and 1.00, a larger penalty parameter C value could achieve lower error rates. Possible explanation for this phenomenon is that since the C value inverse proportionally penalize the misclassified data during the training process, a SVM with a lower C value would tend to separate data groups using hyper-planes with a large margin, causing higher chance of misclassification. On the contrary, a larger C value would encourage the SVM to carefully select the hyper-planes that do not misclassify data points during the training process. By comparing the error rates horizontally between different data dimensionalities, data with higher number of dimensions performed slightly better, which is coherent with the observations in all the previous experiments. In summary, SVM achieved so far the lowest error rate (0.056) among all the first four algorithms, proved to be a powerful classification algorithm.

5 Neural Networks for Classification

Convolutional Neural Networks (CNN) is a special type of neural networks that specialises in image data processing. It is known for using convolution kernels that share the same weights on the whole image/feature map, which made it very efficient and suitable for imagery related tasks.

5.1 Implementation & Experiment

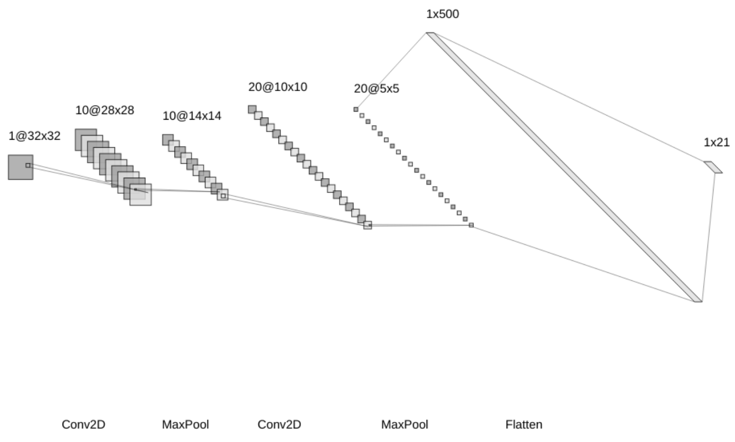

Figure 9: Customized simple CNN structure

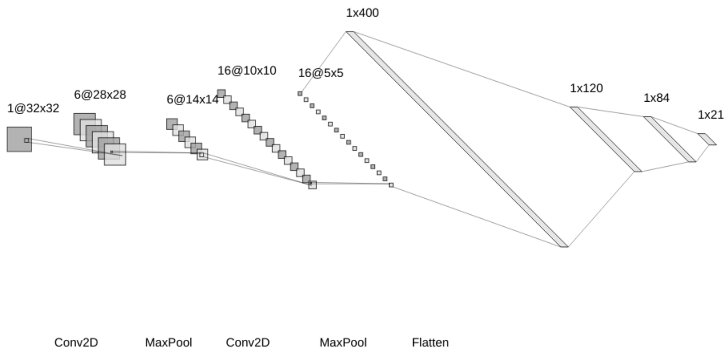

Figure 10: Modified LeNet-5 structure

In this experiment, different from the previous ones, only 20 CMU PIE classes and 1 selfie photo class were used instead of 25 + 1 classes. All the input images were firstly converted to gray-scale, normalized, and then randomly flipped horizontally as a data augmentation technique. A very simple customized CNN model was used in this experiment and its structure is shown in Figure 9. The classification error rate of this simple CNN model after training for 1000 epochs with a batch size of 256 and a learning rate of 3e-4 was 0.317.

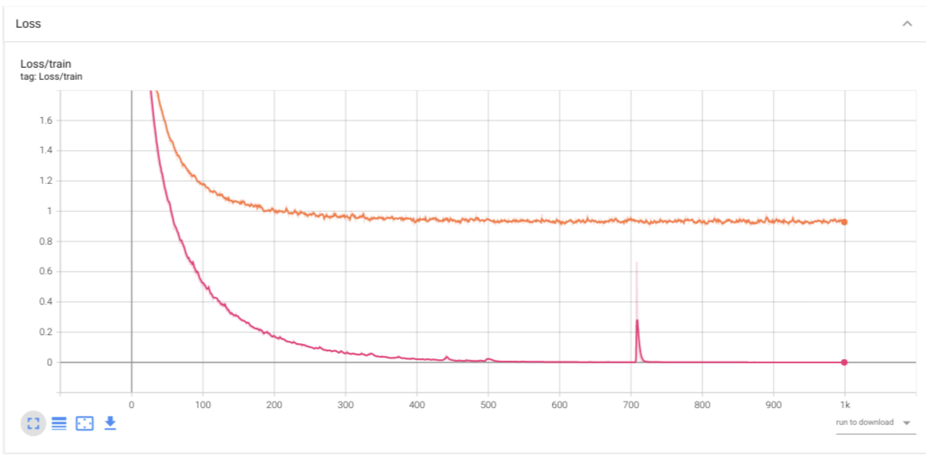

Figure 11: Training loss curves of the simple CNN (orange) and modified LeNet-5 (pink)

5.2 Observations & Discussion

Due to the high classification error rate of the model, an investigation was performed to identify the cause of this huge error. By tabulating the class-wise error rate, it was discovered that among the 20 + 1 classes, only very few classes achieved a accuracy of above 95%, while the rest of the classes only achieved a 0.0% accuracy. Since there are classes which had very high classification accuracy, it might not be caused by the lack of abilities in the feature extraction section (the first half of the network). Based on the network structure, one highly possible cause of the issue might be that during the last stage of the network, the tensor shape was reduced from 500 to 21 too quickly, causing serious loss of information for most of the classes. Thus, a modified version of LeNet-5 (as shown in Figure 10), which has almost the same feature extraction section as the current simple network, but with a more gradual reduction in tensor dimensions, was employed for a comparative study. Under the same training condition and parameters, the modified LeNet-5 achieved a error rate of only 0.0268, with all class-wise accuracy above 90.4% (highests being 100%). For comparison, the loss curves recorded during the training process were shown in Figure 11. In summary, comparing to all the previous experiment results, CNN is so far the most powerful classification method for face recognition in this project.

References

[1] T. Sim, S. Baker, and M. Bsat, "The cmu pose, illumination, and expression (pie) database," in *Proceedings of 5th IEEE International Conference on Automatic Face and Gesture Recognition (FG '02)*, May 2002, pp. 53 – 58.