National University of Singapore

Electrical & Computer Engineering

EE5907 Pattern Recognition

CA2

LUO ZIJIAN

A0224725H

CA2

This report contains six major parts, and each major part is for one question in the assignment. As for the first part of survey, it introduce how to analyse this CA2 .

1 Introduction

In this project, we should design a facial recognition system. In order to better comprehend data distribution, we must use Principal Component Analysis (PCA) to do data dimensionality reduction and visualization. To categorize the face photos, we must also train and use three classification models: Linear Discriminative Analysis (LDA), Support Vector Machine (SVM), and Convolutional Neural Network (CNN).

This project is based on the CMU PIE dataset and ten pictures of my face captured by myself. There are 68 different subjects in all, and I chose 25 randomly from the PIE dataset. I utilize 70 percent of the offered images for training and the remaining 30 percent for testing for each specified subject. My ten selfie shots have also been turned to grayscale and shrunk to the same resolution as the CMU PIE images. And I divided them into seven groups: seven for training and three for testing.

2 Principal Component Analysis (PCA)

This assignment explores Principal Component Analysis (PCA) for compact feature extraction based on data variances. To begin, we should reduce the training and testing images to lower-dimensional feature spaces and use a closest neighbor classifier to classify them.

In order to implement PCA, we should calculate the covariance matrix of data matrix. Firstly, the centered data matrix $X_{d,m}$ shown below:

$$X_{d,n} = \left[(x_1 - \bar{x}) \, (x_2 - \bar{x}) \cdots (x_n - \bar{x}) \right] \tag{1}$$

Next, with centered data matrix, we can compute its SVD, whose D is a diagonal matrix, U and V are orthogonal matrices.

$$X = U_{d,d} D_{d,n} \left(V_{n,n} \right)^T \tag{2}$$

Then we can extract the eigenvector out of the orthogonal matrices. By this way, we can construct the low-dim space especially for PCA

$$x_i = \bar{x} + U_{d,p} U_{d,p}^T \left(x_i - \bar{x} \right) \tag{3}$$

$$U_{d,p} = U(:, 1 : p) \tag{4}$$

We can obtain the eigenvectors of covariance matrix corresponding to the N features and select the largest eigenvalue by this way. And the dimensional reduction of data features is successfully implemented.

2.1 PCA based data distribution visualization

In this part, I will show the visibility of PCA. Here, we should show the 2D, 3D plot and 3 eigenfaces based on the requirement for this CA. In the figures shown below, the yellow points are belonging to my own selfie photos and I implement PCA to reduce the dimension from 1024 to 2 and 3 respectively. You can see from the visualization that points for my selfie photos are closed to each other, so it is pretty easy for us to distinguish them from others.

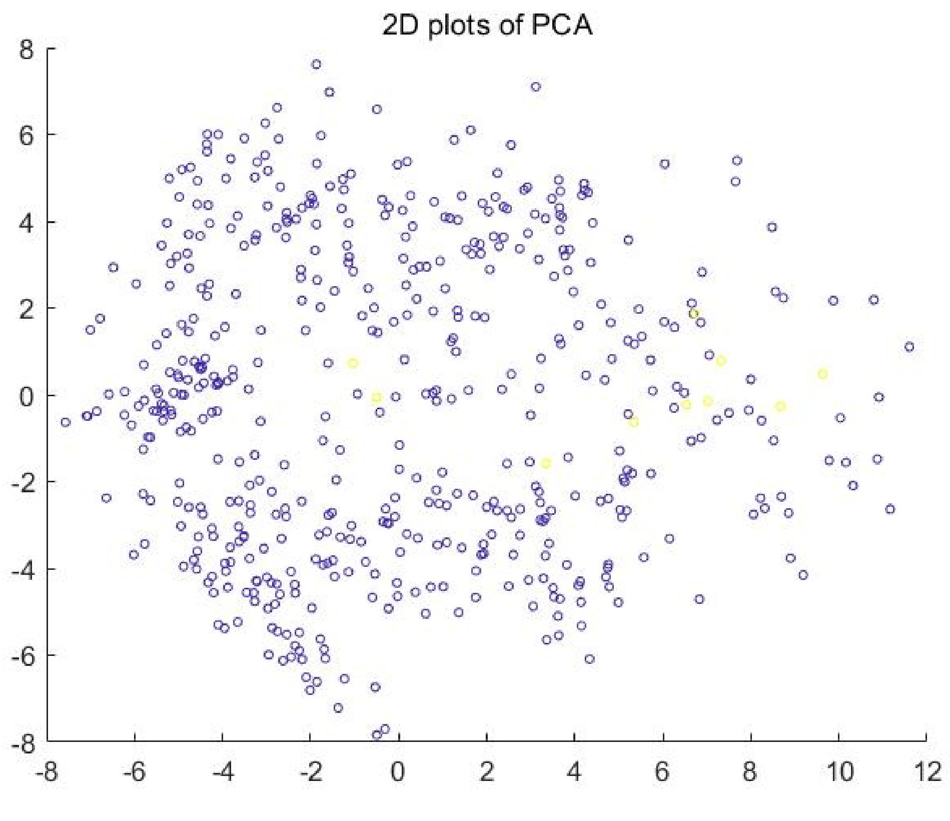


Figure 1: Visualize training set data vectors in 2D for PCA

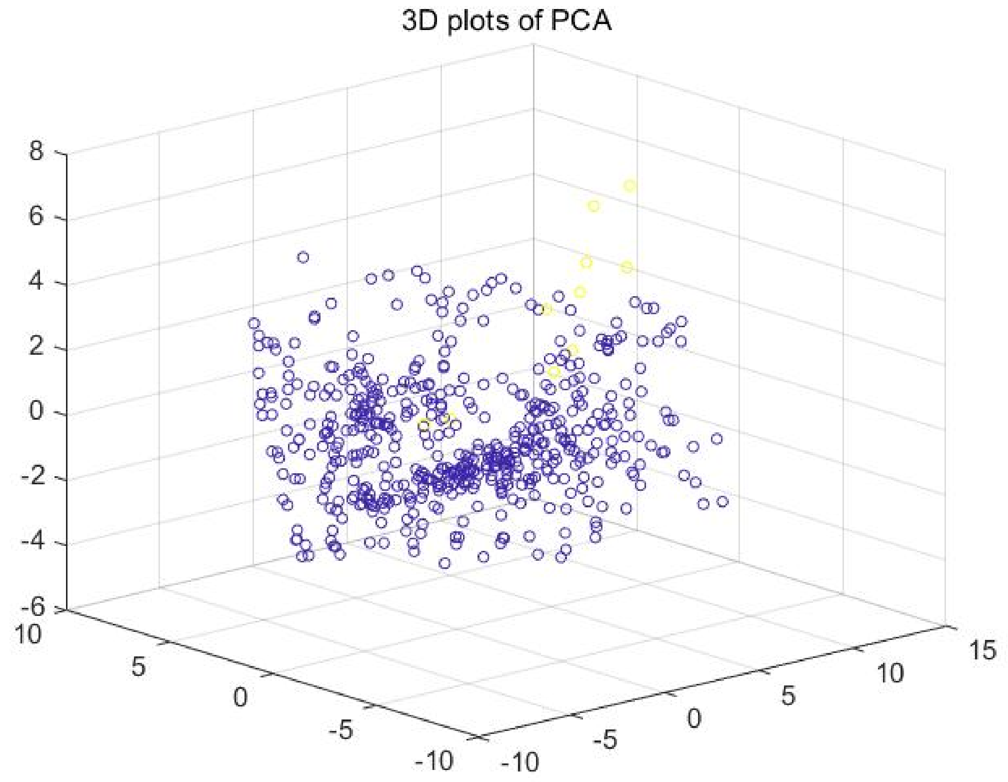


Figure 2: Visualize training set data vectors in 3D for PCA

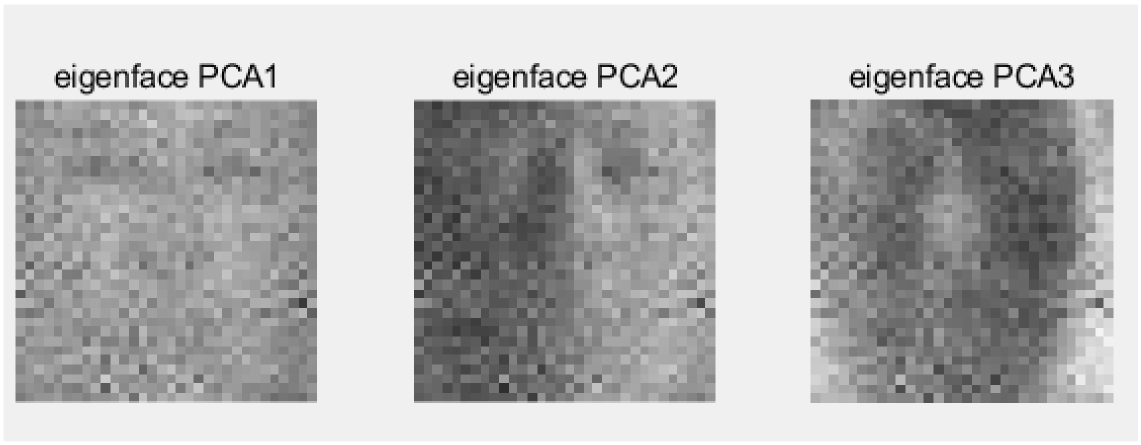


Figure 3: Visualize three eigenfaces

As required, I visualize the corresponding 3 eigenfaces used for the dimensionality reduction in Figure 3

2.2 PCA with nearest neighbor classification results

Nearest neighbor classifier is implemented to classify the test images and output classification accuracy. Here we should compute the Euclidean distance between each training and testing dataset and we should consider the smallest distance as the belonging class. The accuracy when we apply PCA to reduce the dimensionality is shown below Table 1.

	40	80	200
Accuracy of PIE	93.411%	94.823%	95.450%
Accuracy of PIE	100%	100%	100%

Table 1: Accuracy for reducing the dimensionality with PCA method

We can deduce from Table 1 that the accuracy of selfie photographs is 100%. I believe that the reason for this phenomena is that each selfie image is identical to the next. All of the yellow spots representing my selfie photographs in Figure 1 and 2 are also fairly near to each other. As a result, achieving that level of precision is reasonable.

The overall accuracy of the PIE dataset is greater than 90%. The accuracy improves as the number of dimensions increases, as shown in Table 1. The accuracy also improves as the number of training datasets increases.

Furthermore, I believe that the reason PCA with closest neighbor classifier can accurately categorize faces is that we have a large enough training set to feed into the algorithm.

3 Linear Discriminative Analysis (LDA)

By maximizing between-class distance and decreasing within-class distance, Linear Discriminative Analysis (LDA) can determine the most discriminative projection. It is indeed a supervised learning approach that can extract features based on data class membership. As a result, LDA exceeds PCA when it comes to feature classification.

In order to implement LDA algorithm, the first step is to calculate the class-specific mean vector μ_i and covariance matrix S_i by following formulas.

$$\mu_i = \frac{1}{n_i} \sum_{x \in C_i} x \tag{5}$$

$$S_i = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T \tag{6}$$

From the class-specific mean vector μ_i and covariance matrix S_i above, then we can find the within-class scatter matrix S_w and between-class scatter matrix S_B by formulas below.

$$S_W = \sum_{i=1}^{C} \frac{n_i}{N} S_i = \sum_{i=1}^{C} P_i S_i \tag{7}$$

$$S_B = \sum_{i=1}^{C} P_i \left(\mu - \mu_i\right)\left(\mu - \mu_i\right)^T \tag{8}$$

Finally, we can generate the eigenvalue and use SVD to find the eigenvector.

$$[U] = SVD\left(S_W^{-1} S_B\right) \tag{9}$$

We can determine the differences between PCA and LDA from the calculating method. The LDA clearly attempts to model the difference between data classes, whereas the PCA is unable to distinguish between them.

3.1 LDA based data distribution visualization

In this part, I will show the visibility of LDA. Here, we should show the 2D, 3D plot based on the requirement for this CA.

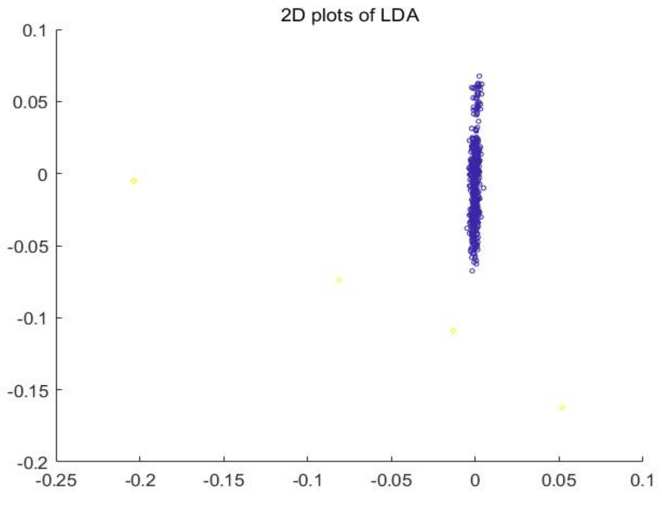


Figure 4: Visualize training set data vectors in 2D for LDA

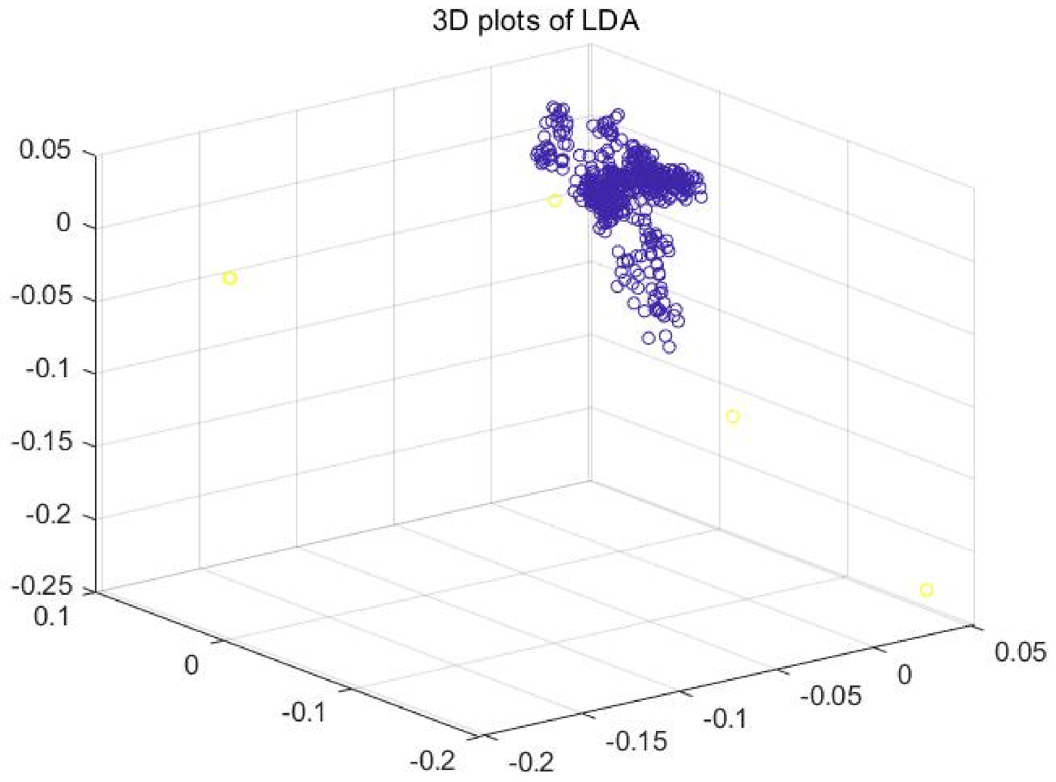


Figure 5: Visualize training set data vectors in 3D for LDA

In the figures shown above, the yellow points are belonging to my own selfie photos and I implement LDA to reduce the dimension from 1024 to 2 and 3 respectively. You can see from the visualization that the yellow points is very far away from blue points. So I think my selfie images can be distinguished easily from the whole dataset and accuracy of my selfie images can be very high.

And we can see the difference between PCA and LDA clearly that LDA can cluster the points more perfectly than PCA. But we cannot just judge that LDA is better than PCA. I think the performance between these two methods is based on the type of data and task requirement.

Nearest neighbor classifier is implemented to classify the test images and output classification accuracy. Here we should compute the Euclidean distance between each training and testing dataset and we should consider the smallest distance as the belonging class. The accuracy when we apply LDA to reduce the dimensionality is shown below Table 2

3.2 LDA with nearest neighbor classification results

Nearest neighbor classifier is implemented to classify the test images and output classification accuracy. Here we should compute the Euclidean distance between each training and testing dataset and we should consider the smallest distance as the belonging class. The accuracy when we apply LDA to reduce the dimensionality is shown below Table 2.

	2	3	9
Accuracy of PIE	24.019%	48.000%	96.941%
Accuracy of PIE	100%	100%	100%

Table 2: Accuracy for reducing the dimensionality with LDA

We can see from Table 2 that the accuracy for selfie photos is 100%, which is similar to PCA. Another thing is that we can classify PIE images more accurately with additional dimensions. LDA has fewer dimensions than PCA, yet the accuracy of the two approaches is comparable. As a result of the different details in algorithm, LDA has a greater classification advantage for some specific tasks.

4 GMM

In this part, the assignment requires me to visualize the test result of GMM with 3 kinds of training data, which are raw face images data and 2 set of pre-processed data by PCA.

4.1 GMM visualization

The Fig 6, 7 and 8 show the clustering results based on raw face image data, data processed by PCA with dimension of 80 and data processed by PCA with dimension of 200 respectively.

In every image below, you can easily find that different coloe represents different cluster. Owing to the difference of training data, the cluster results are not entirely different. In fact, all of them perform well in clustering.

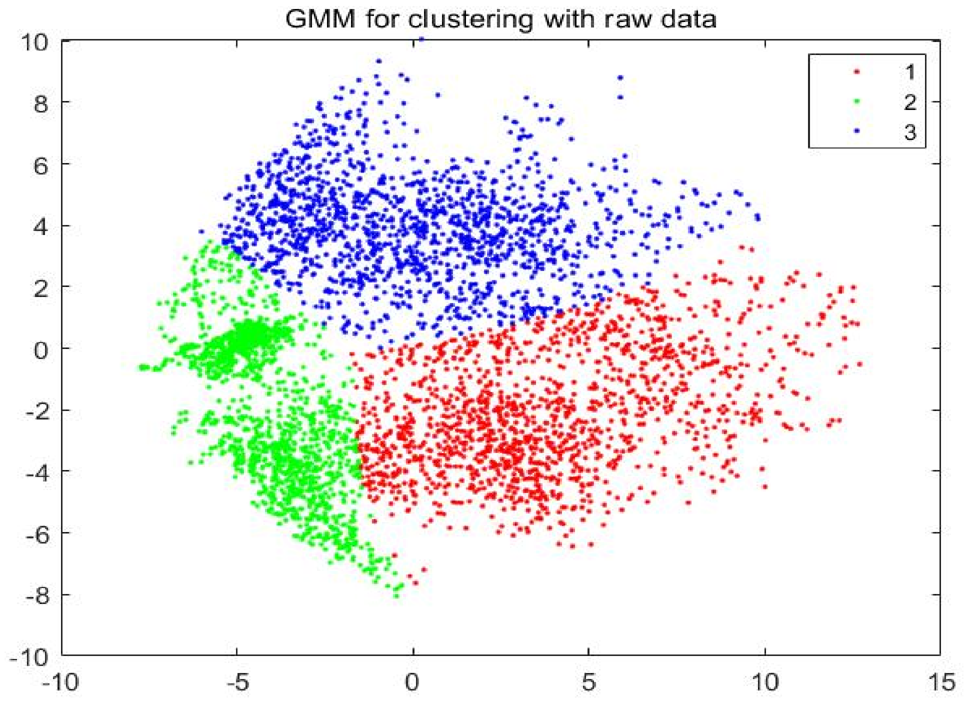


Figure 6: The clustering result of the raw data in GMM

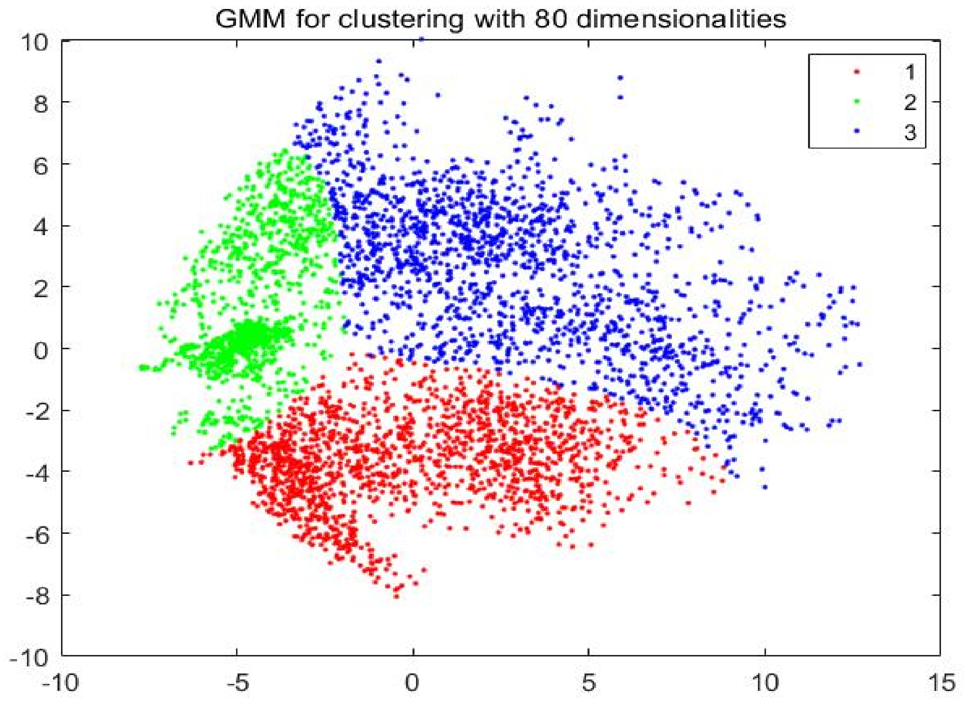


Figure 7: The clustering result of the data with 80 dimensionalities

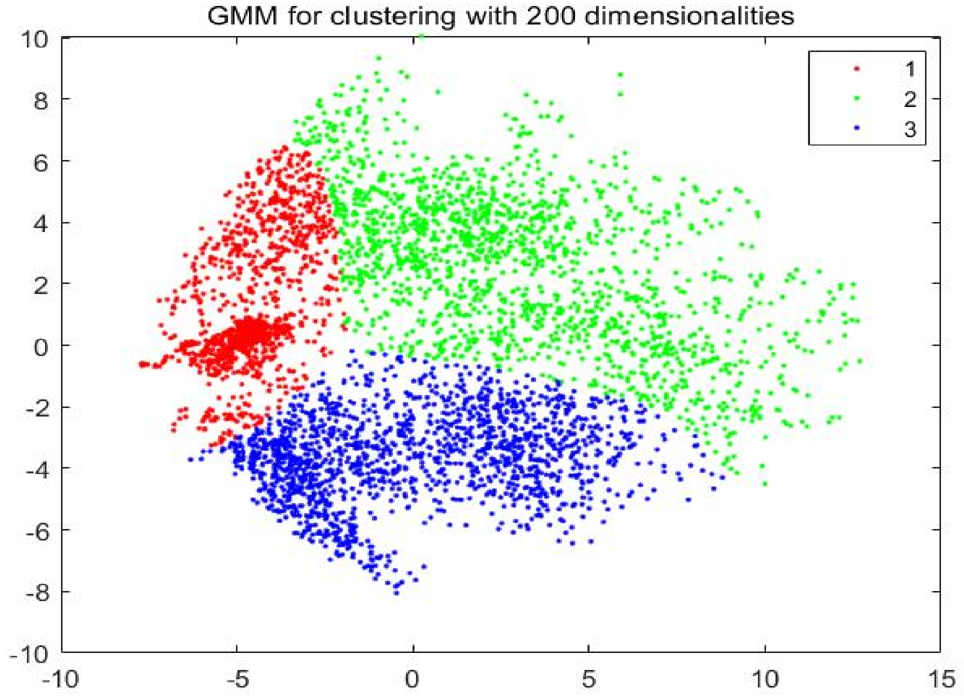


Figure 8: The clustering result of the data with 200 dimensionalities

5 Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a supervised classifier that may increase the margin between classes and increase the breadth of the boundary between them. Therefore, the margin can ensure the best classification results in this scenario.
SVM can generate the optimal superplane based on edge distance, which can be seen in the formular below.

$$W_* = \min_{W,b} \frac{1}{2} W^T W \tag{10}$$

We can think of it as a Quadratic Program (QP) for which there is an efficient global solution algorithm. We can utilize the Penalty parameter to avoid misclassification if the samples are linearly separable.

$$\begin{gathered}W_* = \min_{W,b} \frac{1}{2} W^T W + C \sum_{i=1}^{N} \epsilon_i \\ y_i \left(W^T x_i + b\right) \geq 1 - \epsilon_i\end{gathered} \tag{11}$$

5.1 SVM classification results with different parameter values

In this case, I use SVM to categorize raw face picture data, data treated by PCA with a dimension of 80, and data processed by PCA with a dimension of 200. I'll also experiment with three alternative penalty parameter C values: 0.01, 0.1, and 1.

To begin, I employ a linear kernel function with a variety of training sets and penalty values. Table 5.1 summarizes all of the findings.

	1	0.1	0.01
Accuracy of raw data	85.745%	98.921%	99.333%
Accuracy of 80 dimension	79.745%	95.921%	97.333%
Accuracy of 200 dimesnion	85.441%	98.705%	98.960%

Table 3: Caption

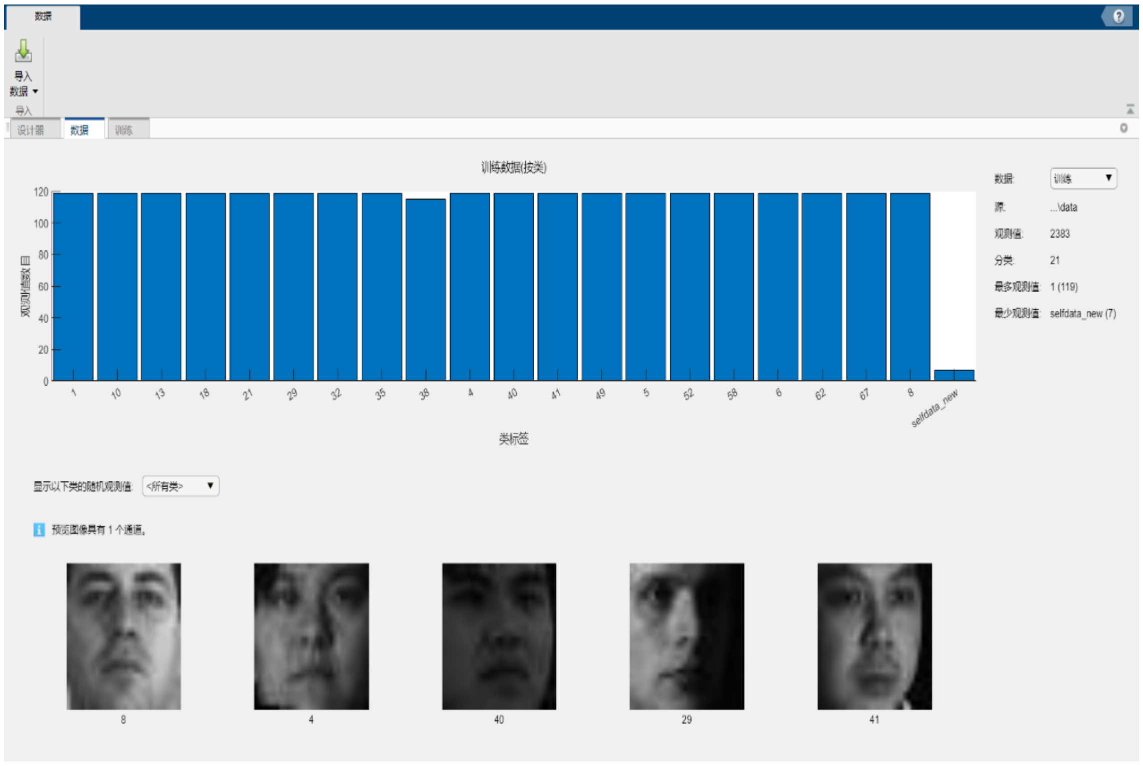
We can observe from the table above that the accuracy increases as the penalty parameter value increases. However, if the penalty is large enough, the classification problem can become overfitting. As a result, we should choose a penalty parameter that strikes a balance between interval size and classification accuracy.

The categorization accuracy can be improved by increasing the dimensions. More dimension implies that we must examine more classification information. The problem becomes easier to solve when the dimension is reduced, but it contains less information. As a result, raw data accuracy can be quite high.

6 Convolutional Neural Network (CNN)

CNNs (Convolutional Neural Networks) are a type of deep, feed-forward artificial neural network that is often used to analyze visual face data.

In this report, I used CNN to analyze the CMU PIE dataset (20 classes) and my selfie photographs (1 class) using an architecture with 20-50-500-21 nodes. To construct it in this section, I utilize the MATLAB toolkit called Deep Network Designer. This is an extremely user-friendly toolkit with a graphical user interface, so I can adjust my CNN structure conveniently. Figure 9 shows the dataset organization.



数据: 训练
源: ...\data
观测值: 2383
分类: 21
最多观测值: 1 (119)
最少观测值: selfdata_new (7)

Figure 9: Load data in CNN

6.1 CNN classification results with different network architectures

Here, we set the convolutional kernel as 5 and each convolution layer is followed by a max pooling layer with a kernel size of 2 and stride of 2. Then the fully connected layer is followed by ReLU. The CNN network architecture is shown in Fig 10. And we can see the detail of this CNN network in Deep Learning Network Analyzer, shown in Fig 11.

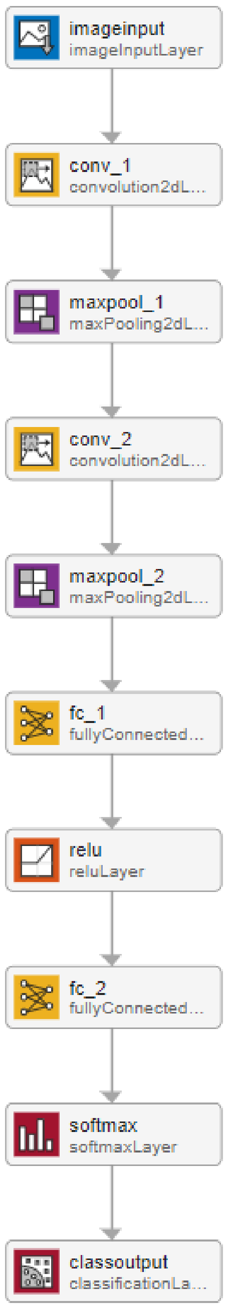
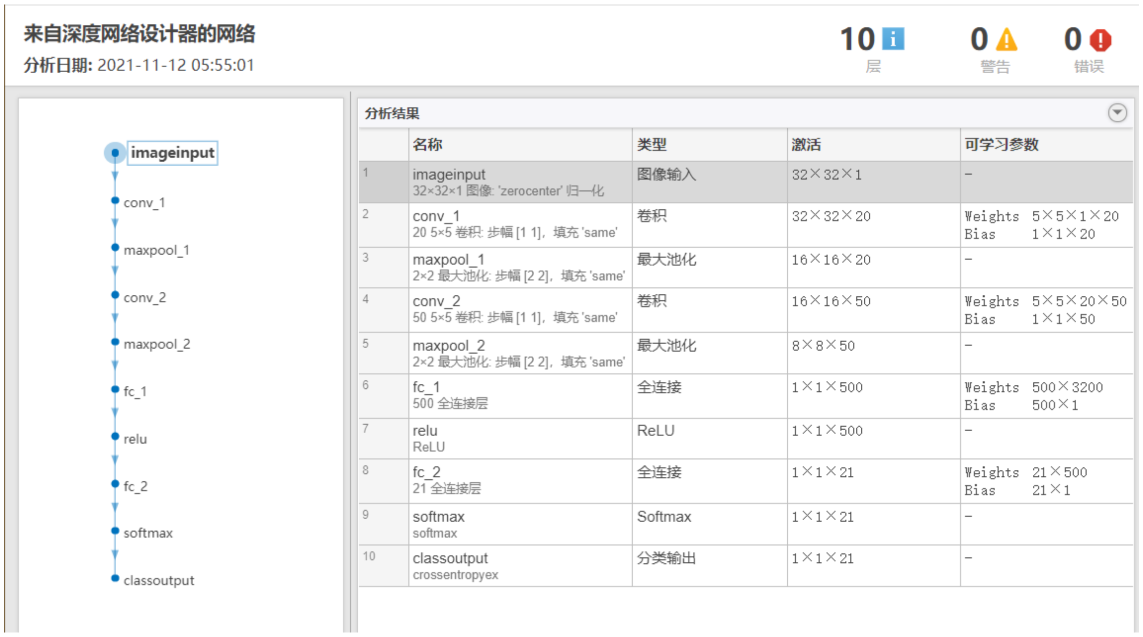


Figure 10: The network design architecture of CNN



来自深度网络设计器的网络
分析日期: 2021-11-12 05:55:01

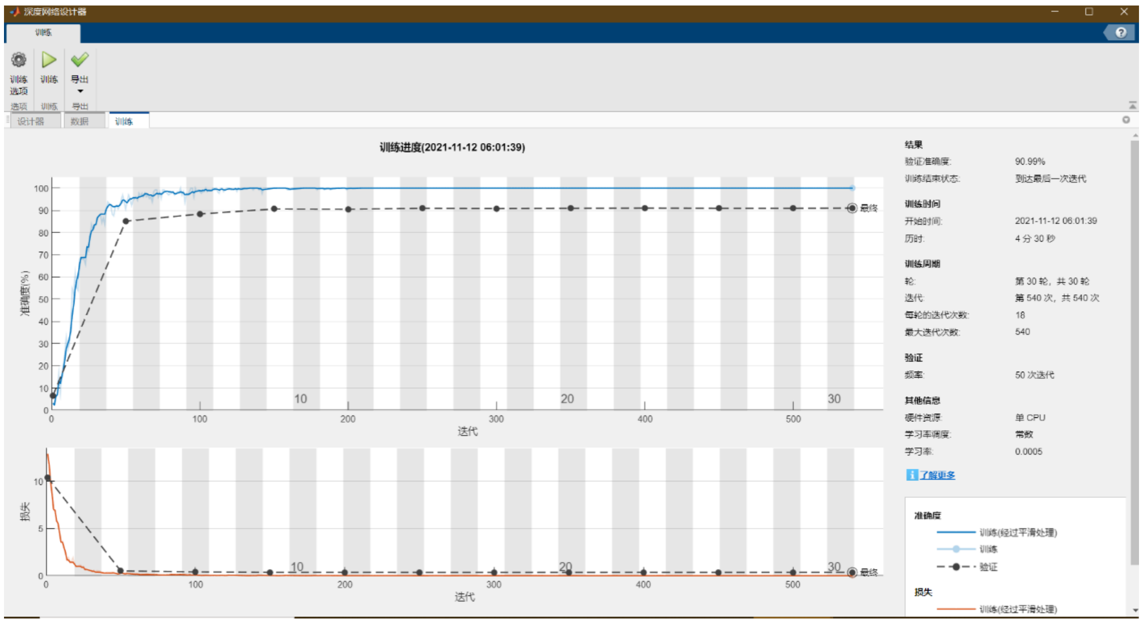
10 层 | 0 警告 | 0 错误

分析结果

	名称	类型	激活	可学习参数
1	imageinput 32×32×1 图像: 'zerocenter' 归一化	图像输入	32×32×1	–
2	conv_1 20 5×5 卷积: 步幅 [1 1], 填充 'same'	卷积	32×32×20	Weights 5×5×1×20 Bias 1×1×20
3	maxpool_1 2×2 最大池化: 步幅 [2 2], 填充 'same'	最大池化	16×16×20	–
4	conv_2 50 5×5 卷积: 步幅 [1 1], 填充 'same'	卷积	16×16×50	Weights 5×5×20×50 Bias 1×1×50
5	maxpool_2 2×2 最大池化: 步幅 [2 2], 填充 'same'	最大池化	8×8×50	–
6	fc_1 500 全连接层	全连接	1×1×500	Weights 500×3200 Bias 500×1
7	relu ReLU	ReLU	1×1×500	–
8	fc_2 21 全连接层	全连接	1×1×21	Weights 21×500 Bias 21×1
9	softmax softmax	Softmax	1×1×21	–
10	classoutput crossentropyex	分类输出	1×1×21	–

Figure 11: The detail of the structure without padding

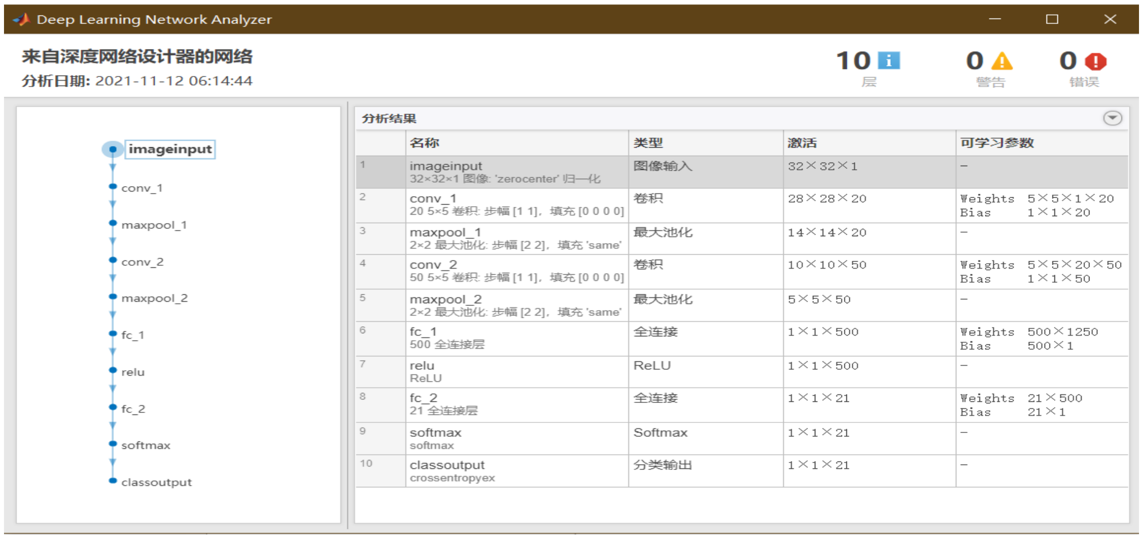
With the structure above, I start my training and finally get the accuracy of this architecture which is 90.99%. Here the learning rate is equal to 0.0005. The outcome shows in Fig 12.



结果	
验证准确度:	90.99%
训练结束状态:	到达最后一次迭代
训练时间	
开始时间:	2021-11-12 06:01:39
历时:	4 分 30 秒
训练周期	
轮:	第 30 轮, 共 30 轮
迭代:	第 540 次, 共 540 次
每轮的迭代次数:	18
最大迭代次数:	540
验证	
频率:	50 次迭代
其他信息	
硬件资源:	单 CPU
学习率调度:	常数
学习率:	0.0005

Figure 12: The accuracy of the structure without padding

Then we add padding into it and the structure is changed, as shown in Fig 13.



来自深度网络设计器的网络

分析日期: 2021-11-12 06:14:44

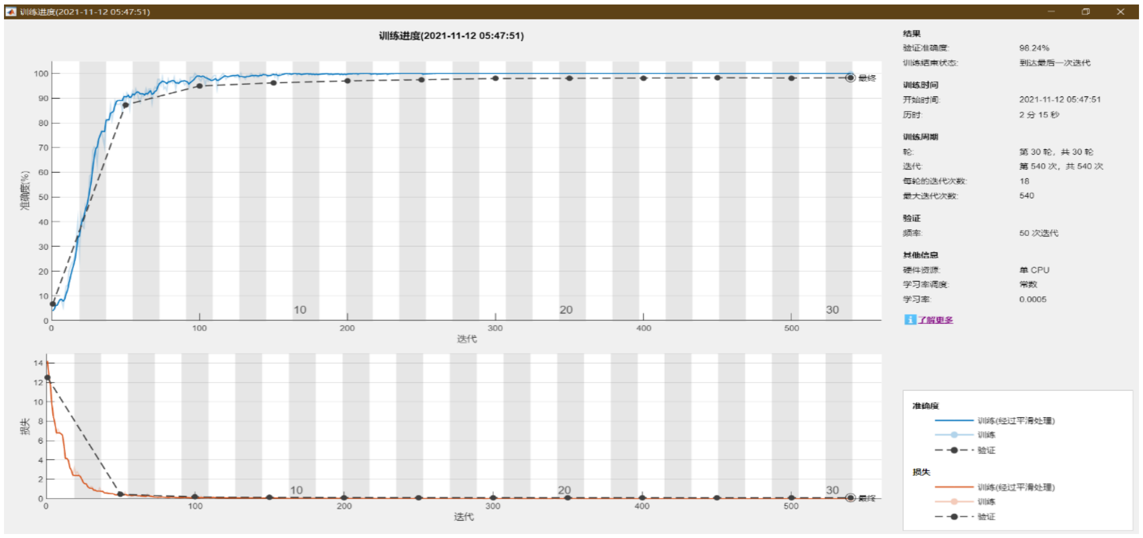
10 层 · 0 警告 · 0 错误

分析结果

	名称	类型	激活	可学习参数
1	imageinput 32×32×1 图像: 'zerocenter' 归一化	图像输入	32×32×1	–
2	conv_1 20 5×5 卷积: 步幅 [1 1]，填充 [0 0 0 0]	卷积	28×28×20	Weights 5×5×1×20 Bias 1×1×20
3	maxpool_1 2×2 最大池化: 步幅 [2 2]，填充 'same'	最大池化	14×14×20	–
4	conv_2 50 5×5 卷积: 步幅 [1 1]，填充 [0 0 0 0]	卷积	10×10×50	Weights 5×5×20×50 Bias 1×1×50
5	maxpool_2 2×2 最大池化: 步幅 [2 2]，填充 'same'	最大池化	5×5×50	–
6	fc_1 500 全连接层	全连接	1×1×500	Weights 500×1250 Bias 500×1
7	relu ReLU	ReLU	1×1×500	–
8	fc_2 21 全连接层	全连接	1×1×21	Weights 21×500 Bias 21×1
9	softmax softmax	Softmax	1×1×21	–
10	classoutput crossentropyex	分类输出	1×1×21	–

Figure 13: The detail of the structure without padding

With the structure above, I start my training and finally get the accuracy of this architecture which is 98.24%, which is better to the structure above. Here the learning rate is equal to 0.0005. The outcome shows in Fig 14.



结果

验证准确度:	98.24%
训练结束状态:	到达最后一次迭代

训练时间

开始时间:	2021-11-12 05:47:51
历时:	2 分 15 秒

训练周期

轮:	第 30 轮，共 30 轮
迭代:	第 540 次，共 540 次
每轮的迭代次数:	18
最大迭代次数:	540

验证

频率:	50 次迭代

其他信息

硬件资源:	单 CPU
学习率调度:	常数
学习率:	0.0005

Figure 14: The accuracy of the structure without padding

Finally, we can conclude that all above CNN structures have a good ability to distinguish the faces images of PIE data and my data.