EE5907 PATTERN RECOGNITION

CA 2 Face Recognition

Name: Yu Shixin

Matriculation Number.: A0195017E

Date:2019/4/23

Data Pre-process

I randomly select 20 out of the 68 different subjects about the CMU PIE dataset, and then disrupt the order of 170 images of each selected PIE. Assigning 1 to 119 as train set and 120 to 170 as test set. Finally, process myface images, and 1 to 7 as train set and 8 to 10 as test set. In the end, I get train set matrix (1024×2387) and test set matrix (1024×1023). This data set save in ***face.mat (for PCA,LDA and SVM)*** and ***faceCNN.mat (for CNN)***.

PCA for Feature Extraction, Visualization and Classification

PCA (Principal Component Analysis) reduces the feature dimension, but keep the main information.

Mathematic:

1. Calculate the covariance matrix S

$$X_{d,n} = [(x_1 - \bar{x}), \dots, (x_n - \bar{x})]$$

$$S = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T$$

2. Compute the eigenvector

   Using SVD to compute the U

$$S = XX^T = UD^2U^T$$

   So the eigenvectors of the S are the columns of U and the eigenvalues are the diagonal elements of D^2.

3. Transform the dataset with PCs

   When reduce the dimension of dataset, we can select eigenvector with highest eigenvalue, and reconstruct the data based the eigenvector.

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

Result:

Firstly, I reduce the dimensionality of vectorized images to 2 and 3, respectively. The figures of the projected data vector in 2D and 3D plots showing as below.

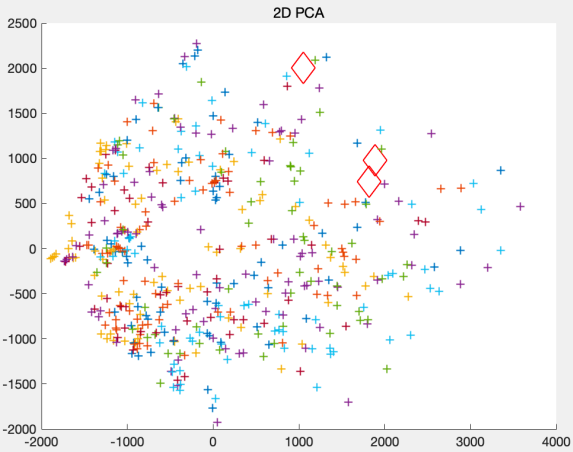

Figure.1 the 2D PCA

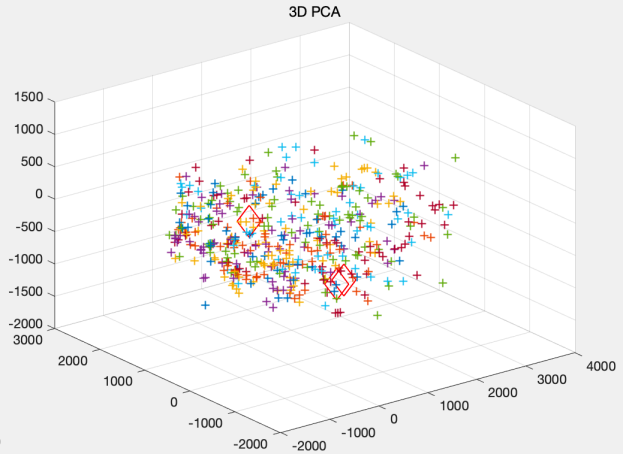

Figure.2 the 3D PCA

For fig.1 and fig.2, the colorful signs (“+”) are those samples with different label, the red hollow diamond represent the projected points corresponding to my photos.

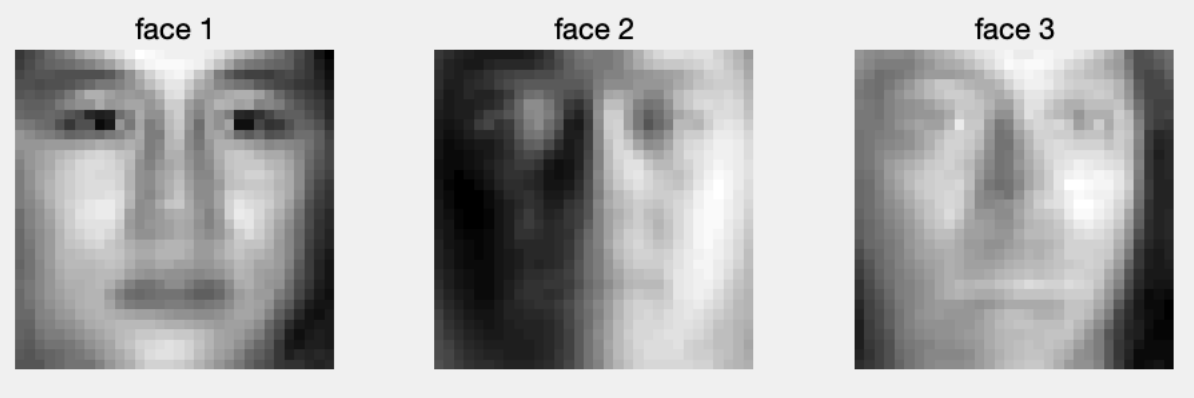

Figure.3 the 3 eigenfaces of dimensionality reduction

According to fig.1 & 2, we can find that same label images can not cluster together, which means the high complexity data (such as face) is difficult to classify by low dimension. From the fig.3, we can find that the first face has the highest similarity to true face image.

Then I apply KNN algorithm to classify and calculate the accuracy on the CMU PIE test images and my own photo. The result is shown as follow:

Table.1 PCA accuracy

Dimension	40	80	200
accuracy of CMU PIE	94.412%	96.275%	97.059%
accuracy of own photo	100%	100%	100%

And we can get the highest accuracy both CMU PIE and own photo with **dimension=259. (97.451% for CMU PIE and 100% for own photo)**. After the 394, the accuracy of CMU PIE keeps stable at 97.451%, but the accuracy of own photo falls to 66.667%).

LDA for Feature Extraction and Classification

LDA (Linear Discriminant Analysis) is another method to reduce the dimension of dataset, which projects the high dimension dataset to the direction that can separate different class data as much as possible.

Mathematic:

1. Calculate the class-specific mean vector (sample) & covariance (scatter) matrix

$$\mu_i = \frac{1}{n_i} \sum_{X \in C_i} X \, , n_i \; is \; the \; size \; of \; class \; C_i$$

$$S_i = \frac{1}{n_i} \sum_{X \in C_i} (X - \mu_i)(X - \mu_i)^T$$

2. Calculate S_W , S_B matrix

$$S_w = \sum_{i=1}^{C} P_i S_i$$

$$C \; is \; the \; class \; number, P_i \; is \; the \; estimate \; of \; the \; prior \; probability \; for \; class \; i$$

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

$$\mu = \frac{1}{N}\sum_{X} X \ , N \ is \ the \ number \ of \ all \ samples$$

3. Find the projection direction

$$W^* = argmax \frac{|W^T S_B W|}{|W^T S_W W|} \rightarrow (S_B - \lambda_i S_W) w_i^* = 0$$

Where $\lambda_i = J(w_i) = scalar$, and the direction W^* is the eigenvector of $S_W^{-1} S_B$

Result:

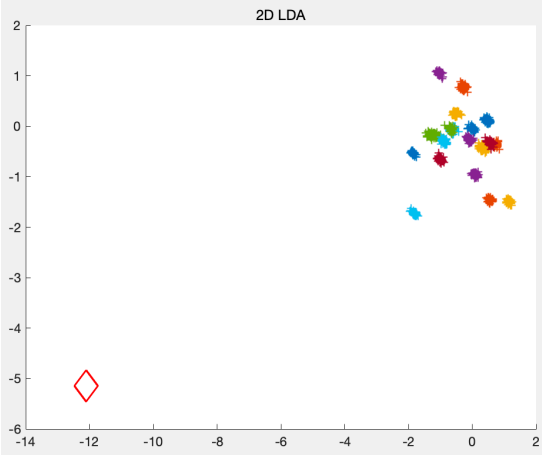

Figure.4 the 2D LDA

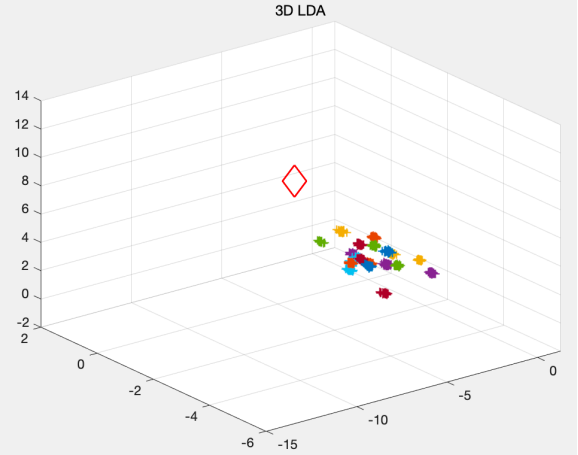

Figure.5 the 3D LDA

According to the fig.4 & 5, we find it is **quite different** with PCA that **same label images cluster together!** The result of LDA seems to be superior to PCA, but LDA can not perform well with small data. Thus, in this experiment, I increase the samples number from 500 to 1100.

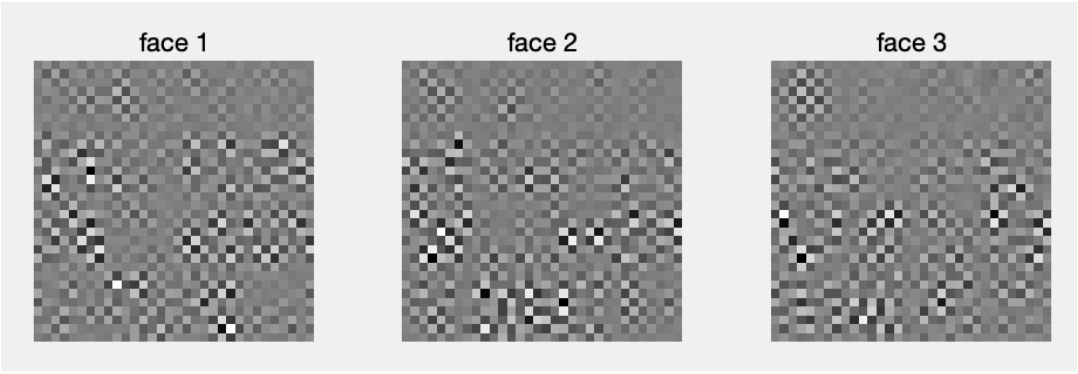

Figure.6 the 3 eigenfaces of dimensionality reduction

Compare Fig.6 with Fig.3, I observe that the eigenvector has less information about the data and the eigenvector of different dimension seems almost the same. It's quite different with the PCA, whose eigenvector is more similar to some class.

Then I apply KNN algorithm to classify and calculate the accuracy on the CMU PIE test images and my own photo. The result is shown as follow:

Table.2 LDA in low dimension

Dimension	**2**	**3**	**9**
accuracy of CMU PIE	37.059%	56.863%	92.941%
accuracy of own photo	33.333%	33.333%	33.333%

Table.3 PCA in low dimension

Dimension	**2**	**3**	**9**
accuracy of CMU PIE	17.647%	28.627%	82.255%
accuracy of own photo	33.333%	100%	100%

Compare tab.2 & 3, we find that in the same low dimension, LDA is superior to PCA when the training data set is large (2387). And we can get the highest accuracy of LDA, when **dimension=516 with 98.725% for CMU PIE and 33.333% for my own photo**. The reason why the accuracy of my own photo can not reach the 100% may that the composition of the photo I took is a little different with the CMU PIE, so the distance is quite far with the CMU PIE.

Thus, we know that when the training data is not small, LDA is better than PCA, even in low dimension, LDA has a higher accuracy for CMU PIE.

SVM for Classification

SVM (Support Vector Machine) is a discriminative classifier with supervised learning to do data binary classification. Its decision boundary is to solve the maximum-margin hyperplane of samples.

Mathematic:

Using SVM to solve problem, the key is maximize the margin, which is shown as below:

$$M = \frac{2}{||W||}$$

The objective function is maximizing the margin, which is equal to minimize the $||W||, or\ ||W||^2 = W^T W$.

Subject to each training point on the correct side (the constraint). There are N constraints we have:

$$W^T X_i + b \geq 1, if\ Y_i = 1$$

$$W^T X_i + b \leq -1, if\ Y_i = -1$$

So we can unify them:

$$Y_i(W^T X_i + b) \geq 1, for\ all\ i$$

There are two solutions for linearly non-separable data, one is soft-margin SVM:

$$\min_{W,b,\varepsilon} \frac{1}{2} W^T W + C \sum_i \varepsilon_i$$

Subject to:

$$Y_i(W^T X_i + b) \geq 1\ ,\ \varepsilon_i \geq 0\ , for\ all\ i$$

Another is kernel SVM:

$$\max_{a_i} \sum a_i - \frac{1}{2} \sum_{i,j} a_i a_j Y_i Y_j \phi(X_i)' \phi(X_j)$$

Subject to:

$$a_i \geq 0\ , for\ all\ i\ , \sum a_i Y_i = 0$$

Result:

Table.4 SVM for CMU PIE

Dimension	C=0.01	C=0.1	C=1
80	99.2157%(1012/1020)	99.2157%(1012/1020)	99.2157%(1012/1020)
200	99.4118%(1014/1020)	99.4118%(1014/1020)	99.4118%(1014/1020)

Table.5 SVM for own photo

Dimension	C=0.01	C=0.1	C=1
80	100%(3/3)	100%(3/3)	100%(3/3)
200	100%(3/3)	100%(3/3)	100%(3/3)

According to table.4 & 5, we can find that comparing dimension and C, the dimension dominates the accuracy. We can get a good performance when dimension=20, about 93.6275%, and the SVM tool does not support too small dimension, such as 10.

And I also try other kernel function with default parameter:

Table.6 different kernel function type

	Linear: u'*v	**polynomial:** $(\gamma u'v + coef)^{degree}$	**RBF:** $e^{-\gamma\|u-v\|^2}$
C=0.01	99.2157%	98.3333%	31.4706%
C=0.1	99.2157%	98.3333%	31.5686%
C=1	99.2157%	98.3333%	19.902%

So according to table.6, we know that linear and polynomial kernel function are more suitable to classify face images, and linear is better.

Neural Networks

In deep learning, a Convolutional Neural Network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. It consists of some convolutional and subsampling layers optionally followed by fully connected layers. The convolution layer extracts feature to form feature maps. The subsampling layers reduce the spatial resolution of each feature map, through which a certain degree of shift and distortion invariance is achieved and the dimensionality of the feature is reduced. Finally, we can use those CNN extracted features to classify data. Here is the normal structure:

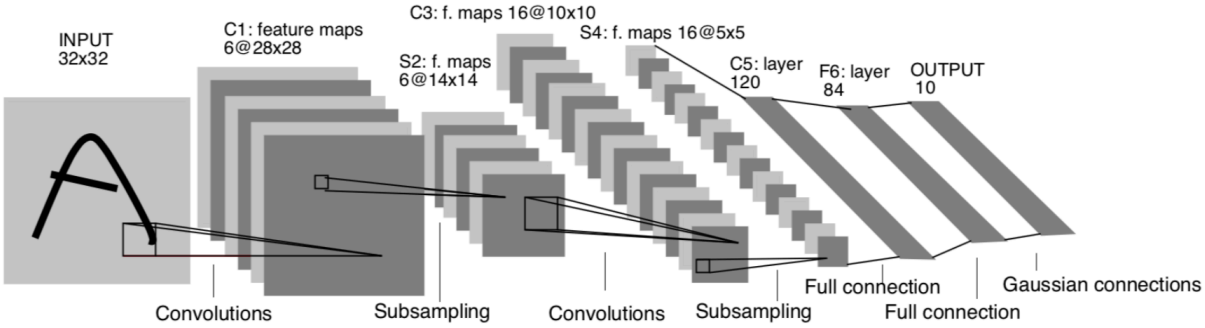

Figure.5 the 3D LDA

Result:

According to the requirement of CA2, this CNN with two convolutional layers and one fully connected layer, with the architecture specified as follows: number of nodes: 20-50-500-21. 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. The result is shown in Fig.6.

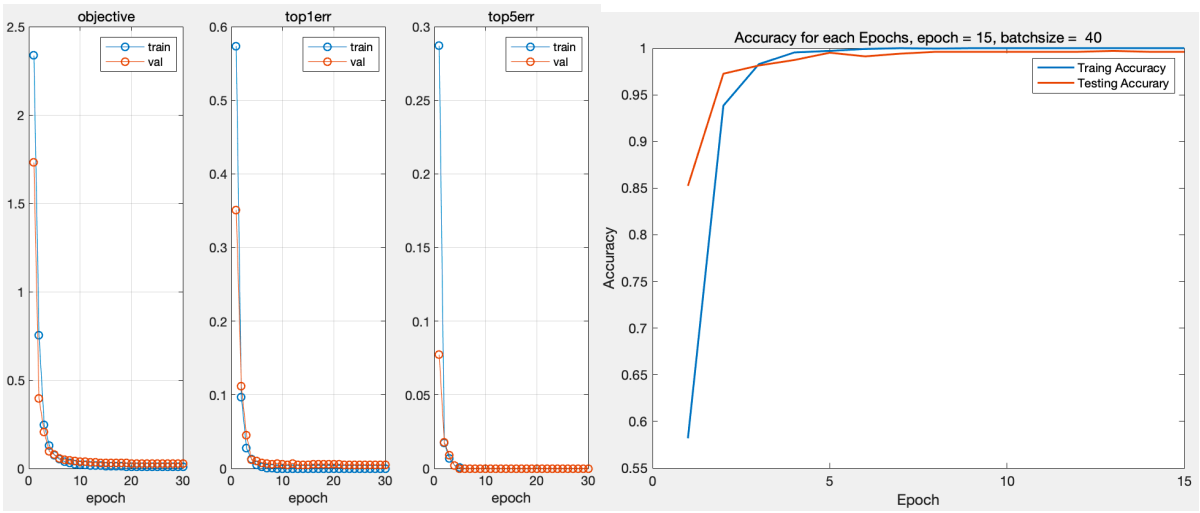

Figure.6 the accuracy of CNN

Because each run with different number of epoch (10/20/30/…) has same accuracy in same epoch, I experiment with epoch = 30, and try different value with some parameters.

Table.7 different BatchSize (epoch=30)

Batch Size	MAX train accuracy	MAX test accuracy
10	100 % (18)	99.7067 % (14)
20	100 % (14)	99.5112 % (20)
30	100 % (10)	99.7067 % (13)
40	**100 % (7)**	**99.7067 % (13)**
50	100 % (9)	99.7067 % (13)
70	100 % (10)	99.609 % (8)
100	100 % (9)	99.5112 % (11)

We can get a good performance at 40 batch size, and the following experiments I will run in this default situation.

Table.8 different learning rate (epoch=15 , batch size=40)

Learning rate	MAX train accuracy	MAX test accuracy
$logspace(-1,-2,100)$	100 % (14)	99.4135 % (14)
$logspace(-1,-4,100)$	100 % (13)	99.3157 % (11)
$logspace(-1,-6,100)$	100 % (13)	99.0225 % (7)
$logspace(-2,-4,100)$	**100 % (7)**	**99.7067 % (13)**
$logspace(-2,-6,100)$	**100 % (7)**	**99.7067 % (13)**
$logspace(-2,-8,100)$	100 % (7)	99.5112 % (7)
$logspace(-3,-4,100)$	99.8743 % (13)	99.5112 % (14)
$logspace(-3,-6,100)$	99.7905 % (13)	99.4135 % (14)

According to table.8, we can get a good result with learning rate between $logspace(-2,-4,100)$ and $logspace(-2,-6,100)$.

From above all, we can build a CNN with 15 epochs, 40 batch sizes and $logspace(-2,-5,100)$.

Summary

According to the four classification algorithm, we can find CNN has the best performance.

PCA	LDA	SVM	CNN
97.451%	98.725%	99.4118%	99.7067%