EE5907 Pattern Recognition

Assignment 2 Face Recognition

SEM 1 AY2019/20

Deadline: 2400 GMT+8, Nov 24, 2019

HAN JIE

A0116448A

Electrical and Computer Engineering

National University of Singapore

November 23, 2019

Introduction

This assignment is a practice to solve face recognition problem using different pattern recognition methods. In first two parts, *Principal Component Analysis* (PCA) and *Linear Discriminant Analysis* (LDA) are used to reduced the dimensionality of the data, followed by *k-Nearest Neighbors* (KNN), first nearest neighbor in this assignment, to classify the faces. The last two parts are to use *Support Vector Machine* (SVM) and *Convolutional Neural Network* (CNN) to classify the faces.

Dataset

The project will be conducted on the *CMU PIE dataset2* and 10 selfie images. There are in total 68 different subjects plus one selfie. The training process is based on 70% of 20 randomly selected subjects and 7 self-taking images while test is based on 30% of the randomly selected subjects and 3 self-taking images. Due to the relatively small training and test size for self-taking images, the classification accuracy for self-taking images are quite low, yet the overall accuracy is satisfying enough with such simple methods.

PCA for Feature Extraction, Visualization and Classification

The size of raw face image is 32×32 pixels, resulting in a 1024 dimensional vector for each image. However, images may not be accurately classified because of the curse dimentionality with the classification method (KNN) we used. In this case, it is necessary to reduce the dimension of the data before passing to the classification model. PCA is a classic unsupervised dimension reduction method.

The basic idea of PCA is to find a new sets of eigenvectors and eigenvalues to represent the data with minimal data loss. Then the new data is reconstructed based on the eignevectors.

Steps:

1. Compute the covariance matrix;
2. Perform eigenvalue decomposition;
3. Output PCs matrix.

Let $\{\vec{x}_i\}$ be a set of N column vectors of dimension D ($D = 1024$). Define the covariance matrix S_x of the dataset as

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

where $\bar{x}$ is the mean of the dataset:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

The d largest principle components are the eigenvectors $\vec{w}_i$ corresponding to the d largest eigenvalues. The eigenvectors of $\mathbf{S}$ can usually be found by using singular value decomposition. The d eigenvectors can also be used to project the data into a d dimensional space. Define

$$\mathrm{W} = [\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_d]$$

The projection of vector $\vec{x}$ is $\vec{y} = \mathrm{W}^T\vec{x}$. The corresponding scatter matrix S_y of the vectors $\{\vec{y}_i\}$ is:

$$\mathrm{S}_y = \mathrm{W}^T\mathrm{S}_x\mathrm{W}$$

When the image data's dimension is reduced to 2, it can be visualized in a 2D plane as shown in Figure 1. Note the red round dots denote self-taking images, which are aggregated together as expected.

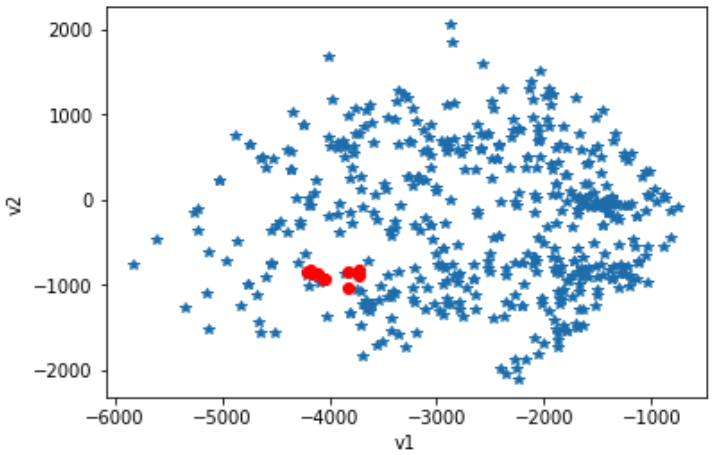

Figure 1: PCA 2D Plot

When the image data's dimension is reduced to 3, it can be visualized in a 3D space as shown in Figure 2. Note the black dots denote self-taking images, which, same as 2D plot, are aggregated together. It also means that PCA has done a great job extracting the features from a high dimensionality dataset.

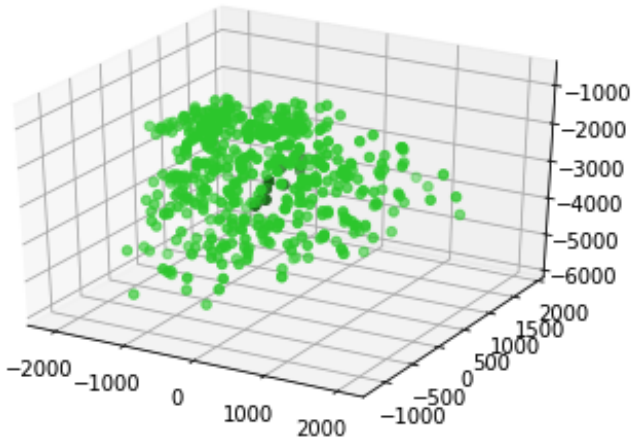

Figure 2: PCA 3D Plot

Even PCA is very effective in dimension reduction, 2D and 3D are too extreme cases, of which the dimension is too low for the face images to be classified. Hence, later in the section, more dimensions are used to feed to KNN classifiers, which does a satisfying job.

By choosing the first 3 eigenvectors, the first 3 eigenfaces can be plotted as Figure 3.

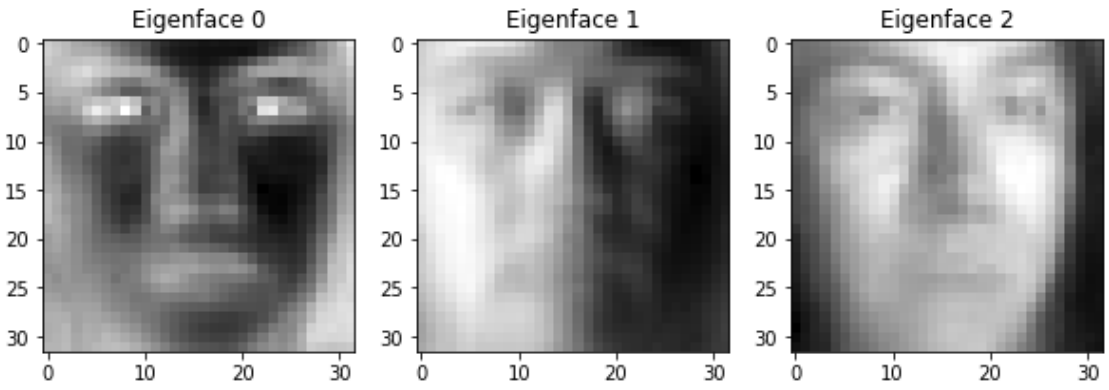

Figure 3: PCA 3 Eigenfaces

Classification Result and Interpretation

KNN classifier from *sklearn* is used for classifying the face images. In this assignment, only the nearest neighbor is considered, i.e. *1-NN* classifier is used.

Table 1 shows the training and test accuracy of CMU PIE dataset based on different dimensions and $1-NN$ classifier.

Accuracy	40D	80D	200D
Train	100%	100%	100%
Test	92.1%	94.1%	95.1%

Table 1: PCA and 1NN Classification Result on CMU PIE

Table 2 shows the training and test accuracy of CMU PIE dataset based on different dimensions and $1-NN$ classifier.

Accuracy	40D	80D	200D
Train	42.9%	42.9%	28.6%
Test	100%	100%	66.7%

Table 2: PCA and 1NN Classification Result on Self-taking Images

The accuracy on self-taking images are quite low on training set, the reason might be the small sample size. 7 training images and 3 test images are considered as very small dataset in pattern recognition. With larger sample size, the accuracy should be improved significantly just like CMU PIE dataset.

LDA for Feature Extraction and Classification

Similar to PCA, LDA is also a classic dimensionality reduction algorithm. However, differetn from PCA, LDA is a supervised algorithm, meaning that the labels of the dataset is considered when reducing the dimension. LDA finds most discriminative projection by maximizing **between-class distance** and minimizing **within-class distance**.

Steps:

1. Compute the mean vector;
2. Compute between-class covariance (scatter) matrix;
3. Compute within-class covariance (scatter) matrix.
4. Perform eigenvalue decomposition;
5. Output projected matrix.

Let $\{\vec{x}_i\}$ be a set of N column vectors of dimension D ($D = 1024$). Define the class-specific covariance matrix S_i of the dataset as

$$\mathrm{S}_i = \frac{1}{n_i} \sum_{x \in C_i}^{N} (x_i - \mu_i)(x_i - \mu_i)^T$$

where μ_i is the class-specific mean vector:

$$\mu_i = \frac{1}{n_i} \sum_{i \in C_i}^{N} x$$

Define the within-class covariance matrix S_W as

$$\mathrm{S}_W = \sum_{i=1}^{C} \frac{n_i}{N} S_i = \sum_{x \in C_i}^{N} P_i S_i$$

Define the between-class covariance matrix S_W as

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

Then the total variance is:

$$S_T = S_W + S_B$$

Since the classification problem is a multiclass dimension reduction, we will new seek $(C-1)$ projections $[y_1, y_2, ..., y_{C-1}]$ by means of $(C-1)$ projection vectors w_i arranged by columns into a project matrix $W = [w_1|w_2|...|w_{C-1}]$:

$$y_i = w_i^T x \Rightarrow y = W^T x$$

Now, the ultimate goal is to find the W that maximises the ratio of between-class to within-class scatters. Denote the ratio as $J(W)$:

$$J(W) = \frac{|S_B|}{|S_W|} = \frac{|W^T S_B W|}{|W^T S_W W|}$$

The optimal projection matrix $W*$ is the one whose columns are the **eigenvectors** corresponding to the largest **eigenvalues**, similar to PCA, denote as:

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

where $\lambda_i = J(w_i) = scalar$

Using eigenvector decomposition or SVM can easily get W, then the projected data can be transformed using W. The 2D and 3D visualisation is shown as figure 4 and figure 5 respectively.

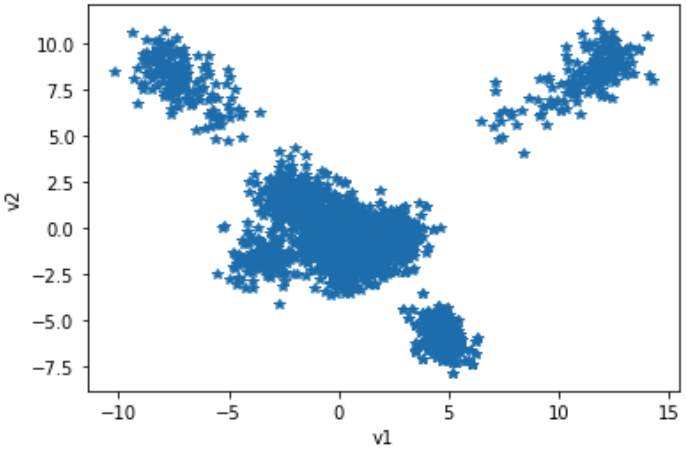

Figure 4: LDA 2D Plot

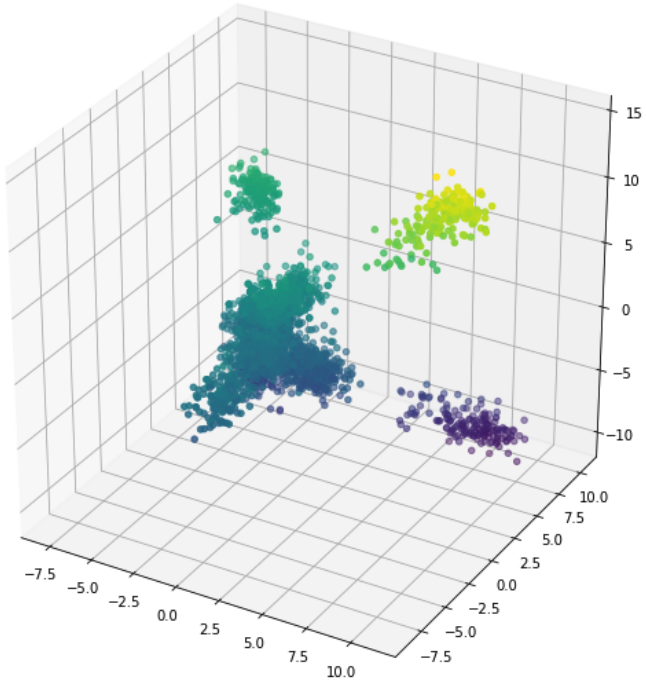

Figure 5: LDA 3D Plot

Results and interpretation

As same previous section, LDA is used for dimensionality reduction and $1-NN$ is used for classification. In this section, the data is transformed into 2D, 3D and 9D data.

Table 3 shows the training and test accuracy of CMU PIE dataset based on different dimensions and $1-NN$ classifier.

Accuracy	2D	3D	9D
Train	100%	100%	100%
Test	64.1%	73.2%	83.9%

Table 3: LDA and 1NN Classification Result on CMU PIE

Table 4 shows the training and test accuracy of CMU PIE dataset based on different dimensions and $1-NN$ classifier.

Accuracy	2D	3D	9D
Train	0%	0%	14.3%
Test	0%	0%	33.3%

Table 4: LDA and 1NN Classification Result on Self-taking Images

LDA does not do a good job in face recognition with 2D, 3D and 9D. Yet, it is not hard to find the trend that the accuracy increases as the dimensionality increases. It is justifiable to have the hypothesis that the low accuracy is due to the extreme low dimensionality. To verify the hypothesis, the dimensions used in PCA is tested, i.e. 40D, 80D and 200D. By increasing the dimensionality moderately, the accuracy is satisfying enough as shown in Table 5.

Accuracy	40D	80D	200D
Train	100%	100%	100%
Test	93.0%	94.8%	96.9%

Table 5: LDA and 1NN Classification Result on CMU PIE with 40D, 80D and 200D

SVM for Classification

Supported Vector Machine (SVM) is a supervised, discriminative classifier. SVM can output an optimal huperplane which categorises samples based on edge distance. The problem can be simplied as:

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

Subject to $y_i(W^T x_i + b) \geq 1$

This problem is known as *Quadratic Program* (QP), for which efficient global solution algorithms exist.

Yet, not all the samples are linearly separable. Penalty parameter C is introduced to have some tolerance to misclassification. The problem becomes:

$$W* = \min_{W,b} \frac{1}{2} W^T W + C \sum_{i=1}^{N} \epsilon_i$$

Subject to $y_i(W^T x_i + b) \geq 1 - \epsilon_i$ for all i, $\epsilon_i \geq 0$ for all i.

The C parameter tells the SVM optimization how much misclassification is penalised on each training sample. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points, because it is less penalised.

The result of using PCA and SVM with 80D and 200D, $C = 0.001, 0.1, 1$ is shown in Table 6.

Accuracy	80D	200D
$C = 0.01$	73.4%	92.2%
$C = 0.1$	70.1%	97.0%
$C = 1$	80.0%	92.8%

Table 6: PCA and SVM Classification Result on CMU PIE

Dimensionality-wise, the higher the dimension, the higher the accuracy. But only 80D and 200D are used for the testing, the specific trend cannot be derived.

Penalty parameter-wise, when the dimension is 80, the accuracy

Neural Networks

Convolutional Neural Network (CNN) is a type of Neural Network with convolutional layers which is used for image feature extraction. As requested in the assignment description, the CNN architecture is shown as Figure 6

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 20)        520
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 20)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 50)        25050
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 50)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1250)              0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               625500
_________________________________________________________________
dense_2 (Dense)              (None, 21)                10521
=================================================================
Total params: 661,591
Trainable params: 661,591
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: LDA 2D Plot

There are many parameters choices and parameters that can be fine-tuned in this CNN architecture. For the loss function, *categorical crossentropy* is used since this is a multiclass

classification problem. The learning rate of optimizer is also fine-tuned to be a small number 1×10^{-4}. The detail implementation is shown in next subsection.

Implementation in Python using Keras

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten
from keras import optimizers
import keras.backend as K

K.clear_session()
model = Sequential()
# add first layer - 20
model.add(Conv2D(20, (5, 5), activation='relu', input_shape=(image_size,
    image_size, 1)))
# add max pooling layer
model.add(MaxPool2D(pool_size=(2, 2)))
# add first layer - 20
model.add(Conv2D(50, (5, 5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
# flatten the image from 32 * 32 to 1024 * 1
model.add(Flatten())
# add first layer - 20
model.add(Dense(500, activation='relu'))
# add first layer - 21 (20 subject + 1 selfie subject)
model.add(Dense(21, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['accuracy'])

model.fit(X_train, y_train_onehot_encoded, batch_size=128,
          epochs=50, verbose=1, validation_split=0.3)
model.evaluate(X_test, y_test_onehot_encoded)
```

Result

CNN is training for 50 epoches with *batch size=128*, the result shows the classification accuracy to be 96.0% as shown in Figure 7.

```
model.evaluate(X_test, y_test_onehot_encoded)
1023/1023 [==============================] - 1s 887us/step
[0.18061967968999, 0.9599217978157722]
```

Figure 7: CNN Result