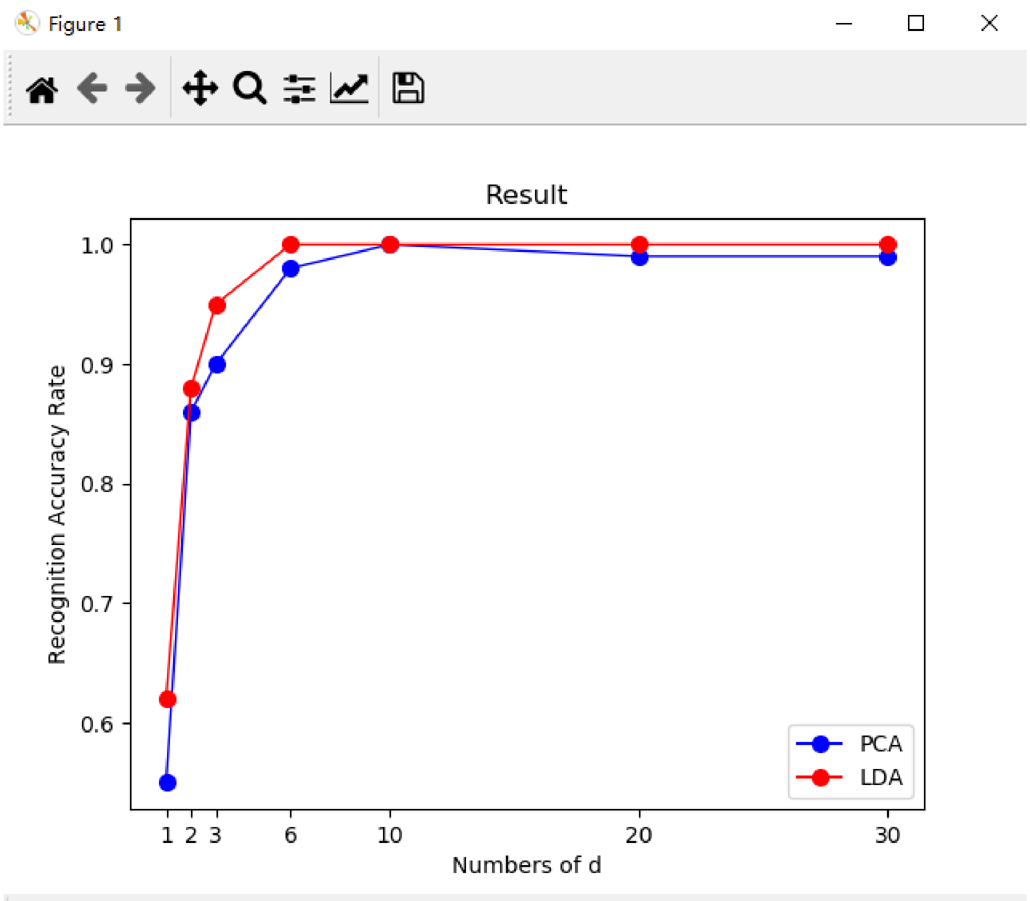
COEN 240 Machine Learning

Homework #6

Name: Jinhao Wang   ID: 4302178

Problem1:



```
PCA Recognition Accuracy Rate =  [0.55, 0.86, 0.9, 0.98, 1.0, 0.99, 0.99]
 LDA Recognition Accuracy Rate =  [0.62, 0.88, 0.95, 1.0, 1.0, 1.0, 1.0]
```

The result shows that LDA has a better accuracy rate than PCA. This is probably because LDA is supervised learning, who will take training data labels into model training, while PCA is unsupervised learning, it only uses plain training data.

Attachment:

Problem 1 Code:

```
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from matplotlib import pyplot as plt
from matplotlib.lines import Line2D
from matplotlib.image import imread
import numpy as np
import random

d = [1, 2, 3, 6, 10 ,20 ,30]
width = 92
height = 112
img_set = np.ndarray(shape=(100, height*width))     #whole data set
train_set = np.ndarray(shape=(80, height*width))    #store training data
test_set = np.ndarray(shape=(20, height*width))     #store test data
train_label = []    #store which subject the training data belongs to
test_label = []     #store which subject the test data belongs to
pca_res = [0, 0, 0, 0, 0, 0, 0]     #final PCA result
tmp_pca = []                        #PCA accuracy of one experiment
lda_res = [0, 0, 0, 0, 0, 0, 0]     #final LDA result
tmp_lda = []                        #LDA accuracy of one experiment

#read images and split them into training set and test set, store training and test
labels
def generate_set():
    train_counter = 0
    test_counter = 0
    index_set = random.sample(range(0, 9), 2)
    train_label.clear()
    test_label.clear()
    for i in range(10):
        for j in range(10):
            img = plt.imread('s'+str(i+1)+'/'+str(j+1)+'.pgm')
            if j in index_set:
                test_set[test_counter,:] = np.array(img).flatten()
                test_counter += 1
                test_label.append(i)
            else:
                train_set[train_counter,:] = np.array(img).flatten()
                train_counter += 1
                train_label.append(i)
```

```python
def cal_PCA():
    #each experiment, clear tmp_pca array
    tmp_pca.clear()
    #use different number of principal components
    for i in range(7):
        pca0 = PCA(n_components=d[i])
        pca0_operator = pca0.fit(train_set)
        train_reduced = pca0_operator.transform(train_set)
        test_reduced = pca0_operator.transform(test_set)

        correct_res = 0
        for index, sample in enumerate(test_reduced):
            idx = np.argmin(np.linalg.norm(sample - train_reduced, axis=1))
            if test_label[index] == train_label[idx]:
                correct_res += 1

        tmp_pca.append(correct_res / len(test_reduced))

def cal_LDA():
    #each experiment, clear tmp_lda array
    tmp_lda.clear()
    #use PCA to reduce dimensionality first
    pca0 = PCA(n_components=40)
    pca0_operator = pca0.fit(train_set)

    train_reduced = pca0_operator.transform(train_set)
    test_reduced = pca0_operator.transform(test_set)
    #use different number of principal components
    for i in range(7):
        lda =  LinearDiscriminantAnalysis(n_components=d[i])
        lda_operator = lda.fit(train_reduced,train_label)

        train_projection = lda_operator.transform(train_reduced)
        test_projection = lda_operator.transform(test_reduced)

        correct_res = 0
        for index, prediction in enumerate(test_projection):
            idx = np.argmin(np.linalg.norm(train_projection - prediction, axis=1))
            if test_label[index] == train_label[idx]:
                correct_res += 1

        tmp_lda.append(correct_res / len(test_projection))
```

```
#20 independent experiments
for l in range(20):
    generate_set()
    cal_PCA()
    cal_LDA()
    for k in range(7):
        pca_res[k] += tmp_pca[k]
        lda_res[k] += tmp_lda[k]

#final result and plot
for k in range(7):
    pca_res[k] = round(pca_res[k]/20, 2)
    lda_res[k] = round(lda_res[k]/20, 2)

print("PCA Recognition Accuracy Rate = ", pca_res, "\n", "LDA Recognition Accuracy Rate
= ", lda_res)

X, Y1, Y2 = d, pca_res, lda_res
plt.plot(X, Y1, color= 'b', linewidth = 1, marker='o', markerfacecolor='b',
markersize=7)
plt.plot(X, Y2, color= 'r', linewidth = 1, marker='o', markerfacecolor='r',
markersize=7)

legend = [Line2D([0], [0], marker='o', color='b', label='PCA', markerfacecolor='b',
markersize=7),
          Line2D([0], [0], marker='o', color='r', label='LDA', markerfacecolor='r',
markersize=7)]

plt.legend(handles=legend, loc='lower right')
plt.xticks(X)
plt.xlabel('Numbers of d')
plt.ylabel('Recognition Accuracy Rate')
plt.title('Result')
plt.show()
```