Homework 4

Collaborators:

Name: Shen Kai

Student ID: 21921071

Problem 4-1. Spectral Clustering

In this problem, we will try a dimensionality reduction based clustering algorithm – Spectral Clustering.

(a) We will first experiment Spectral Clustering on synthesis data

Answer:

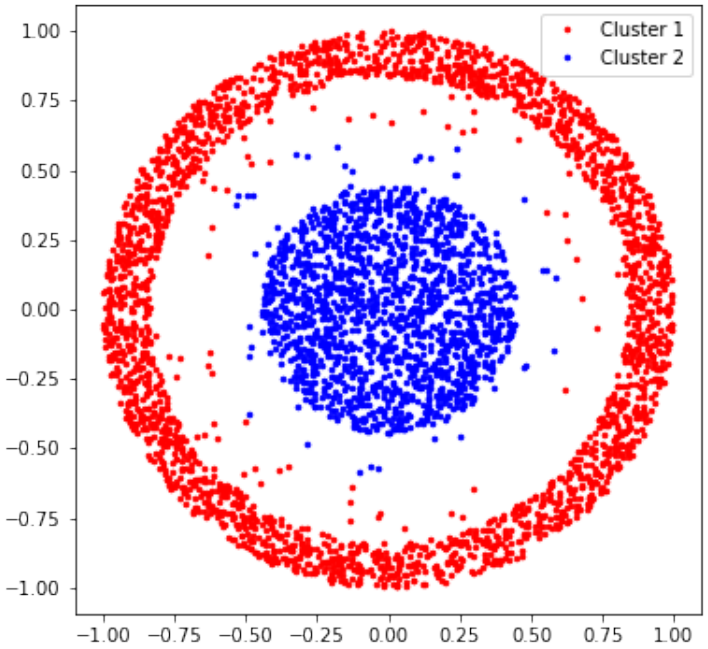

Figure 1: The spectral clustering results.

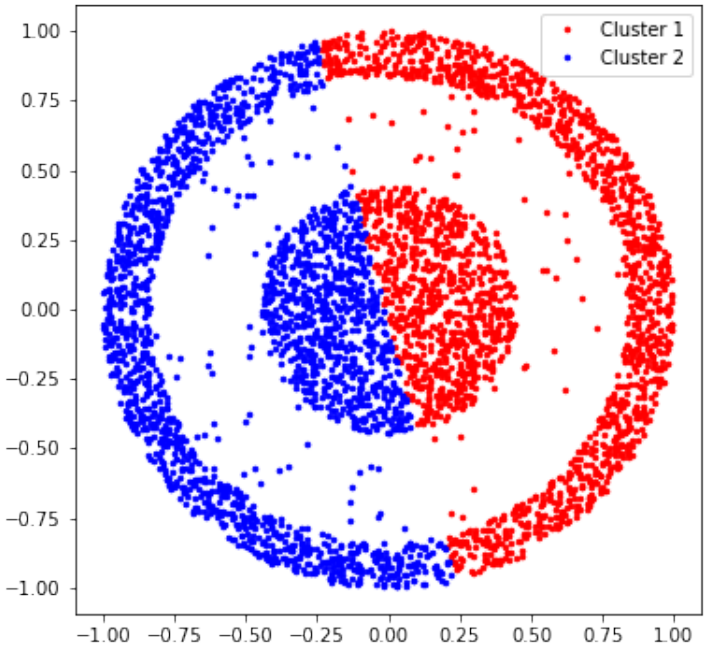

Figure 2: The kmeans results.

(b) Now let us try Spectral Clustering on real-world data.

Answer:

```
spectral accuracy: 0.5700341167551175
spectral mutual info: 0.5689778019765478
kmeans accuracy: 0.4841736163760425
kmeans mutual info: 0.29615827392821137
```

Figure 3: The experiment on real data.

Problem 4-2. Principal Component Analysis Let us deepen our understanding of PCA by the following problems.

(a) Your task is to implement *hack_pca.m* to recover the rotated CAPTCHA image using PCA.

Answer: In this part, we assume that the main direction of the picture is $v = [x_1, x_2, x_3]$. Then $\hat{v} = A * v = [1, 0, 0]$. It will fill the main components of the PCA's vector. So the angle will be $angle = arctan(\frac{x_1}{x_2})$

The results are shown as following.

```
In [33]: from hack_pca import hack_pca

         # Hack different images yourself
         img = hack_pca('1.gif')

         plt.imshow(img)

Out[33]: <matplotlib.figure.AxesImage at 0x121935908>
```

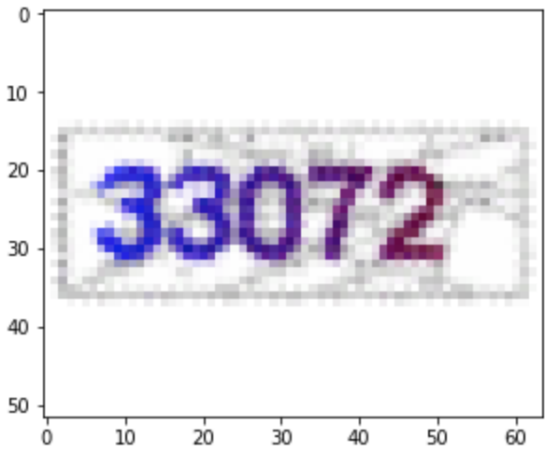

Figure 4: The rotated picture by PCA.

(b) Now let us apply PCA to a face image dataset.

Answer: (i) The results are shown as following:

```
# 2. Run PCA
# begin answer
from pca import PCA
print(fea_Train.shape, "-----")
w, v = PCA(fea_Train)
print(v.shape)
# end answer
# 3. Visualize eigenface
# begin answer
show_face(v.T)
# end answer
```

```
(200, 1024) -----
(1024, 1024)
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 5: The eigenface.

(ii)

The PCA:

The error rate by different dim of PCA are shown as following:

```
pca error rate: 0.38
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 6: The error rate of dim=8.

```
pca error rate: 0.29
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 7: The error rate of dim=16.

```
pca error rate: 0.26
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 8: The error rate of dim=32.

```
pca error rate: 0.25
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 9: The error rate of dim=64.

```
pca error rate: 0.235
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 10: The error rate of dim=128.

The recovered face:

```
pca error rate: 0.38
original pic:
recover with PCA from dim = 8
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

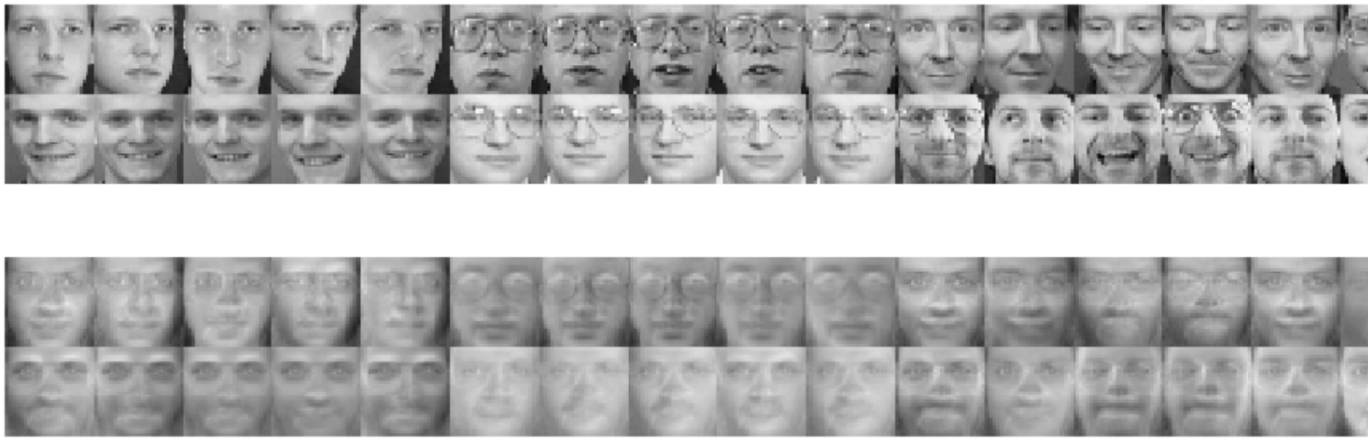

Figure 11: The recovered face of dim=8.

```
pca error rate: 0.29
original pic:
recover with PCA from dim = 16
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 12: The recovered face of dim=16.

```
pca error rate: 0.26
original pic:
recover with PCA from dim = 32
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 13: The recovered face of dim=32.

```
pca error rate: 0.25
original pic:
recover with PCA from dim = 64
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning: C
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 14: The recovered face of dim=64.

```
pca error rate: 0.235
original pic:
recover with PCA from dim = 128
```

```
/Users/Alan/workspace/homework/ml/hw4/ml2019fall_hw4/pca/show_face.py:21: ComplexWarning:
eal discards the imaginary part
  = fea[i * numPerLine + j, :].reshape((faceH, faceW)).transpose()
```

Figure 15: The recovered face of dim=128.

The LDA:

In [39]:
```
from LDA import LDA

# Your code here
# begin answer
v, w = LDA(fea_Train, gnd_Train)
dim = 8
V = v[:, 0:dim]
fea_train_reduce = np.matmul(fea_Train, V)
fea_test_reduce = np.matmul(fea_Test, V)
# end answer

# 5. Run KNN in low dimensional space
# begin answer
from knn import knn
y_ret = knn(fea_test_reduce, fea_train_reduce, gnd_Train, 2)
error = np.sum((gnd_Test != y_ret)) / len(y_ret)
print("LDA error rate: {}".format(error))
# end answer
```

```
LDA error rate: 0.13
```

Figure 16: The error rate of dim=8.

In [40]:
```
from LDA import LDA

# Your code here
# begin answer
v, w = LDA(fea_Train, gnd_Train)
dim = 16
V = v[:, 0:dim]
fea_train_reduce = np.matmul(fea_Train, V)
fea_test_reduce = np.matmul(fea_Test, V)
# end answer

# 5. Run KNN in low dimensional space
# begin answer
from knn import knn
y_ret = knn(fea_test_reduce, fea_train_reduce, gnd_Train, 2)
error = np.sum((gnd_Test != y_ret)) / len(y_ret)
print("LDA error rate: {}".format(error))
# end answer
```

```
LDA error rate: 0.06
```

Figure 17: The error rate of dim=16.

In [41]:
```
from LDA import LDA

# Your code here
# begin answer
v, w = LDA(fea_Train, gnd_Train)
dim = 32
V = v[:, 0:dim]
fea_train_reduce = np.matmul(fea_Train, V)
fea_test_reduce = np.matmul(fea_Test, V)
# end answer

# 5. Run KNN in low dimensional space
# begin answer
from knn import knn
y_ret = knn(fea_test_reduce, fea_train_reduce, gnd_Train, 2)
error = np.sum((gnd_Test != y_ret)) / len(y_ret)
print("LDA error rate: {}".format(error))
# end answer
```

```
LDA error rate: 0.055
```

Figure 18: The error rate of dim=32.

In [42]:
```
from LDA import LDA

# Your code here
# begin answer
v, w = LDA(fea_Train, gnd_Train)
dim = 64
V = v[:, 0:dim]
fea_train_reduce = np.matmul(fea_Train, V)
fea_test_reduce = np.matmul(fea_Test, V)
# end answer

# 5. Run KNN in low dimensional space
# begin answer
from knn import knn
y_ret = knn(fea_test_reduce, fea_train_reduce, gnd_Train, 2)
error = np.sum((gnd_Test != y_ret)) / len(y_ret)
print("LDA error rate: {}".format(error))
# end answer
```

```
LDA error rate: 0.055
```

Figure 19: The error rate of dim=64.

In [43]:
```
from LDA import LDA

# Your code here
# begin answer
v, w = LDA(fea_Train, gnd_Train)
dim = 128
V = v[:, 0:dim]
fea_train_reduce = np.matmul(fea_Train, V)
fea_test_reduce = np.matmul(fea_Test, V)
# end answer

# 5. Run KNN in low dimensional space
# begin answer
from knn import knn
y_ret = knn(fea_test_reduce, fea_train_reduce, gnd_Train, 2)
error = np.sum((gnd_Test != y_ret)) / len(y_ret)
print("LDA error rate: {}".format(error))
# end answer
```

```
LDA error rate: 0.055
```

Figure 20: The error rate of dim=128.