# Homework 3 - Dimensionality Reduction

Hovni Singh CS 383 02/24/2022

## 1 Theory Questions

### (a)

Data:

$$\begin{bmatrix} -2 & 1 \\ -5 & -4 \\ -3 & 1 \\ 0 & 3 \\ -8 & 11 \\ -2 & 5 \\ 1 & 0 \\ 5 & -1 \\ -1 & -3 \\ 6 & 1 \end{bmatrix}$$

Calculate mean:

1st column mean:

$$\frac{(-2+-5+-3+0+-8+-2+1+5+-1+6)}{10} = -0.9$$

2nd column mean:

$$\frac{(1+-4+1+3+11+5+0+-1+-3+1)}{10} = 1.4$$

Calculate Standard Deviation:

1st column std:

$$(-2+0.9)^2+(-5+0.9)^2+(-3+0.9)^2+(0+0.9)^2+(-8+0.9)^2+(-2+0.9)^2+(1+0.9)^2+(5+0.9)^2+(-1+0.9)^2+(6+0.9)^2 = 160.90$$

$$std = \sqrt{\frac{160.90}{10-1}} = 4.22$$

2nd column std:

$$(1+1.4)^2+(-4+1.4)^2+(1+1.4)^2+(3+1.4)^2+(11+1.4)^2+(5+1.4)^2+(0+1.4)^2+(-1+1.4)^2+(-3+1.4)^2+(1+1.4)^2 = 164.40$$

$$std = \sqrt{\frac{164.40}{10-1}} = 4.27$$

Standardized matrix = data-mean/std:

$$\begin{bmatrix} -0.2602 & -0.0936 \\ -0.9697 & -1.2635 \\ -0.4967 & -0.0936 \\ 0.2129 & 0.3744 \\ -1.6792 & 2.2462 \\ -0.2602 & 0.8423 \\ 0.4494 & -0.3276 \\ 1.3954 & -0.5615 \\ -0.0237 & -1.0295 \\ 1.6319 & -0.0936 \end{bmatrix}$$

Calculate Covariance Matrix:

$$cov = \frac{w^T w}{N-1}$$

$$\begin{bmatrix} -0.2602 & -0.9697 & -0.4967 & 0.2129 & -1.6792 & -0.2602 & 0.4494 & 1.3954 & -0.0237 & 1.6319 \\ -0.0936 & -1.2635 & -0.0936 & 0.3744 & 2.2462 & 0.8423 & -0.3276 & -0.5615 & -1.0295 & -0.0936 \end{bmatrix} * \begin{bmatrix} -0.2602 & -0.0936 \\ -0.9697 & -1.2635 \\ -0.4967 & -0.0936 \\ 0.2129 & 0.3744 \\ -1.6792 & 2.2462 \\ -0.2602 & 0.8423 \\ 0.4494 & -0.3276 \\ 1.3954 & -0.5615 \\ -0.0237 & -1.0295 \\ 1.6319 & -0.0936 \end{bmatrix} * \frac{1}{9} =$$

$$\begin{bmatrix} 1 & -0.408 \\ -0.408 & 1 \end{bmatrix}$$

Find Eigenvalues:

$$|A - \lambda I| = 0$$

$$|A - \lambda I| = \begin{vmatrix} a11 - \lambda & a12 \\ a21 & a22 - \lambda \end{vmatrix} = (a11 - \lambda)(a22 - \lambda) - a12a21 = \lambda^2 - \lambda(a11 + a22) + (a11a22 - a12a21) = 0$$

$$\lambda^2 - \lambda(1 + -0.408) + ((1)(-0.408) - (1)(-0.408)) = 0$$

$$\lambda = 0.592, 1.408$$

Find Eigevectors: For each value of $\lambda$, solve

$$(A - \lambda I)x = 0$$

$$\begin{bmatrix} 1 & -0.408 \\ -0.408 & 1 \end{bmatrix} - \begin{bmatrix} 0.592 & 0 \\ 0 & 0.592 \end{bmatrix} * \begin{bmatrix} x1 \\ x2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -0.408 \\ -0.408 & 1 \end{bmatrix} - \begin{bmatrix} 1.408 & 0 \\ 0 & 1.408 \end{bmatrix} * \begin{bmatrix} x1 \\ x2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

### (b)

Largest Eigenvalue = 1.408

$$\begin{bmatrix} -0.2602 & -0.0936 \\ -0.9697 & -1.2635 \\ -0.4967 & -0.0936 \\ 0.2129 & 0.3744 \\ -1.6792 & 2.2462 \\ -0.2602 & 0.8423 \\ 0.4494 & -0.3276 \\ 1.3954 & -0.5615 \\ -0.0237 & -1.0295 \\ 1.6319 & -0.0936 \end{bmatrix} * \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1667 \\ -0.2938 \\ 0.4031 \\ 0.1615 \\ 3.9254 \\ 1.1025 \\ -0.7769 \\ -1.9569 \\ -1.0058 \\ -1.7255 \end{bmatrix}$$

## 2 Dimensionality Reduction via PCA

In [258…

```python
from sklearn.datasets import fetch_lfw_people
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np
from math import *
```

In [259…

```python
people = fetch_lfw_people(min_faces_per_person=20, resize=0.7)
image_shape = people.images[0].shape
fig,axes = plt.subplots(2, 5, figsize=(15, 8),
                        subplot_kw={'xticks': () , 'yticks': ()})
for target, image, ax in zip(people.target, people.images, axes.ravel()):
    ax.imshow(image, cmap=cm.gray)
    ax.set_title(people.target_names[target])

print(people.images.shape)
print(len(people.target_names))
```

```
(3023, 87, 65)
62
```


In [260…

```python
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
counts = np.bincount(people.target)
for i, (count, name) in enumerate(zip(counts, people.target_names)):
    print('{0:25} {1:3}'.format(name, count), end = ' ')
    if (i+1) % 3 == 0:
        print()
mask = np.zeros(people.target.shape, dtype=bool)
for target in np.unique(people.target):
    mask[np.where(people.target==target)[0][:50]] = 1

X_people = people.data[mask]
y_people = people.target[mask]
X_people = X_people/255
X_train, X_test, y_train, y_test = train_test_split(X_people, y_people,
                                                    stratify=y_people, random_state=0)
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train,y_train)
```

```
print()
print('Test set score of 1-nn: {:.2f}'.format(knn.score(X_test,y_test)))
```

```
Alejandro Toledo           39 Alvaro Uribe               35 Amelie Mauresmo            21
Andre Agassi               36 Angelina Jolie             20 Ariel Sharon               77
Arnold Schwarzenegger      42 Atal Bihari Vajpayee       24 Bill Clinton               29
Carlos Menem               21 Colin Powell              236 David Beckham              31
Donald Rumsfeld           121 George Robertson           22 George W Bush             530
Gerhard Schroeder         109 Gloria Macapagal Arroyo    44 Gray Davis                 26
Guillermo Coria            30 Hamid Karzai               22 Hans Blix                  39
Hugo Chavez                71 Igor Ivanov                20 Jack Straw                 28
Jacques Chirac             52 Jean Chretien              55 Jennifer Aniston           21
Jennifer Capriati          42 Jennifer Lopez             21 Jeremy Greenstock          24
Jiang Zemin                20 John Ashcroft              53 John Negroponte            31
Jose Maria Aznar           23 Juan Carlos Ferrero        28 Junichiro Koizumi          60
Kofi Annan                 32 Laura Bush                 41 Lindsay Davenport          22
Lleyton Hewitt             41 Luiz Inacio Lula da Silva  48 Mahmoud Abbas              29
Megawati Sukarnoputri      33 Michael Bloomberg          20 Naomi Watts                22
Nestor Kirchner            37 Paul Bremer                20 Pete Sampras               22
Recep Tayyip Erdogan       30 Ricardo Lagos              27 Roh Moo-hyun               32
Rudolph Giuliani           26 Saddam Hussein             23 Serena Williams            52
Silvio Berlusconi          33 Tiger Woods                23 Tom Daschle                25
Tom Ridge                  33 Tony Blair                144 Vicente Fox                32
Vladimir Putin             49 Winona Ryder               24
Test set score of 1-nn: 0.23
```

In [286…

```
k = 1
i = 0
predictions = []
distances = []

for test in X_test:
    i = 0
    distances = []
    for train in X_train:
        dis = np.sqrt(np.sum((train-test)**2))
        distances.append((dis,y_train[i]))
        i +=1
    l = [x[1] for x in sorted(distances)[:1]]
    labels = {}
    if l[0] not in labels:
        labels[l[0]] = 1
    else:
        labels[l[0]] += 1
    predictions.append(max(labels, key=labels.get))

score = np.sum(predictions == y_test)/len(y_test)
print('1-nn score using personal KNN algo:', score)
```

```
1-nn score using personal KNN algo: 0.23255813953488372
```

In [262…

```
m = np.mean(X_train, axis=0)
s = np.std(X_train, axis=0, ddof=1)
sX_train = (X_train-m)/s

sX_test = (X_test-m)/s

cov = (sX_train.T@sX_train)/(len(sX_train)-1)


W,V = np.linalg.eig(cov)
```

In [264…

```
pairs=[(np.abs(W[i]),V[:,i]) for i in range(len(W))]
pairs.sort()
pairs.reverse()
data_pca_100D = np.array([])


data_pca_100D = []

for i in range(0,100):
    data_pca_100D.append(pairs[i][1].reshape(len(pairs[0][1])))
data_pca_100D = np.asarray(data_pca_100D)

data_pca_100D = data_pca_100D.T

new_data = sX_train.dot(data_pca_100D)


Ztest = sX_test.dot(data_pca_100D)
```

In [271…

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(new_data, y_train)
print('Test set 1-nn score using built in KNeighborsClassifier: {:.2f}'.format(knn.score(Ztest,y_test)))

predictions = []
distances = []
dis = 0
l = []

for test in Ztest:
    i = 0
    distances = []
    for train in new_data:
        dis = np.sqrt(np.sum((train-test)**2))
        distances.append((dis,y_train[i]))
        i +=1
    l = [x[1] for x in sorted(distances)[:1]]
    labels = {}
    if l[0] not in labels:
        labels[l[0]] = 1
    else:
        labels[l[0]] += 1
    predictions.append(max(labels, key=labels.get))
```

```
Test set 1-nn score using built in KNeighborsClassifier: 0.25
```

In [272…

```
score2 = np.sum(predictions == y_test)/len(y_test)
```

```
print("100D PCA data 1-nn score using personal KNN algo:", score2)
```

```
100D PCA data 1-nn score using personal KNN algo: 0.25387596899224807
```

In [273…

```
U, s, Vt = np.linalg.svd(sX_train)
```

In [274…

```
data_pca_100D = np.asarray(data_pca_100D)

gon = np.diag(1. / np.sqrt(W[:100]))
white = np.dot(np.dot(data_pca_100D, gon), data_pca_100D.T)
x_white = np.dot(sX_train, white)
```

In [276…

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(x_white, y_train)
print('Test set score of 1-nn with whitened 100D data using built in KNeighborsClassifier: {:.2f}'.format(knn.score(sX_test,y_test)))
print()


predictions = []
distances2 = []
dis = 0
l = []

for test1 in sX_test:
    distances2 = []
    i = 0
    for train in x_white:
        dis = np.sqrt(np.sum((train-test1)**2))
        distances2.append((dis,y_train[i]))
        i+=1
    l = [x[1] for x in sorted(distances2)[:1]]
    labels = {}
    if l[0] not in labels:
        labels[l[0]] = 1
    else:
        labels[l[0]] += 1
    predictions.append(max(labels, key=labels.get))

score = np.sum(predictions == y_test)/len(y_test)
print("100D PCA whitened data 1-nn score using personal KNN algo:", score)
```

```
Test set score of 1-nn with whitened 100D data using built in KNeighborsClassifier: 0.32

100D PCA whitened data 1-nn score using personal KNN algo: 0.32170542635658916
```

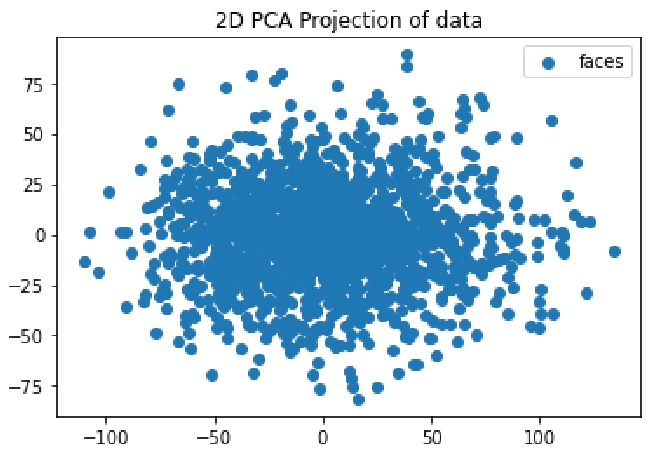
In [277…

```
pairs=[(np.abs(W[i]),V[:,i]) for i in range(len(W))]
pairs.sort()
pairs.reverse()

data_pca = np.hstack((pairs[0][1].reshape(len(pairs[0][1]),1), pairs[1][1].reshape(len(pairs[0][1]),1)))

transformed_data = data_pca.T @ sX_train.T
xy_PCA_2D = np.vstack((transformed_data, y_train)).T
```

## Part 2: The visualization of the PCA result, KNN accuracies

In [279…

```
plt.scatter(xy_PCA_2D[:,0], xy_PCA_2D[:,1], label="faces")
plt.title("2D PCA Projection of data")
plt.legend()

plt.show()
```

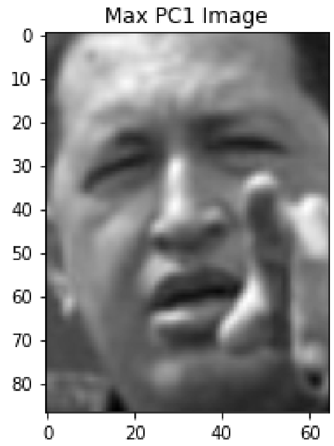


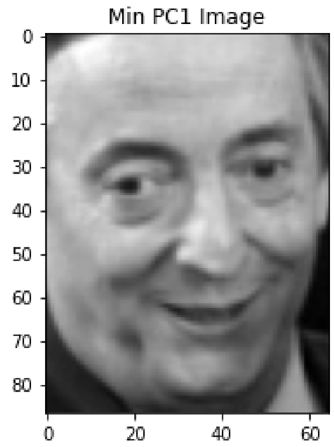

# 3 Eigenfaces

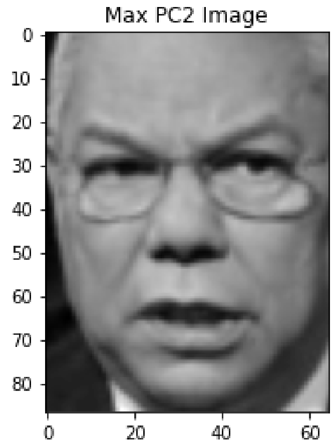
In [281…

```
indexes = np.argsort(W)[::-1]
vals = W[indexes]
vecs = V[:,indexes]


ppc = vecs[:,0:1]
ppc2 = vecs[:,1:2]


x_hat = sX_train @ ppc[:,0].T
x_hat2 =  sX_train @ ppc2[:,0].T

max_pc1 = np.argmax(x_hat)
min_pc1 = np.argmin(x_hat)


max_pc2 = np.argmax(x_hat2)
min_pc2 = np.argmin(x_hat2)


org_people = X_people*225

fullMaxPC1 = org_people[max_pc1]
fullMinPC1 = org_people[min_pc1]
```

```python
fullMaxPC2 = org_people[max_pc2]
fullMinPC2 = org_people[min_pc2]

new_img1 = np.reshape(fullMaxPC1, (87,65))
new_img2 = np.reshape(fullMinPC1, (87,65))

new_img3 = np.reshape(fullMaxPC2, (87,65))
new_img4 = np.reshape(fullMinPC2, (87,65))

plt.imshow(new_img1, cmap=cm.gray)
plt.title('Max PC1 Image')
plt.show()

plt.imshow(new_img2, cmap=cm.gray)
plt.title('Min PC1 Image')
plt.show()

plt.imshow(new_img3, cmap=cm.gray)
plt.title('Max PC2 Image')
plt.show()

plt.imshow(new_img4, cmap=cm.gray)
plt.title('Min PC2 Image')
plt.show()
```





In [282…

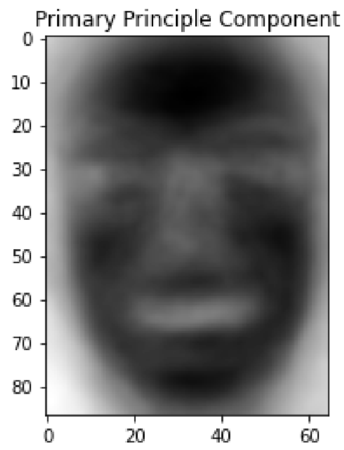
```python
visual = np.reshape(ppc, (87,65))

zVal = X_train @ ppc

x_hat_reconstruct = zVal @ ppc.T


iml = x_hat_reconstruct[0,:]


alpha = 0.95
n = np.diag(V)
tot = np.sum(abs(n))
check_k = 0

for i in range (0,len(n)):
    check_k = check_k + n[int(indexes[i])]
    if (check_k / tot) >= alpha:
        break
```

```
k = i


components = vecs[:,0:k]

k_z = X_train @ components

k_x_hat_reconstruct = k_z @ components.T


iml2 = k_x_hat_reconstruct[0,:]
```

### i. Visualization of primary principle component

In [283…

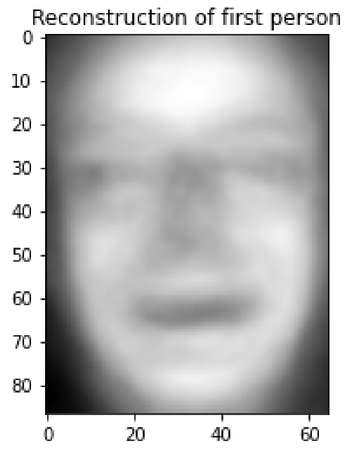
```
plt.imshow(visual, cmap=cm.gray)
plt.title('Primary Principle Component')
plt.show()
```




### ii. Number of principle components needed to represent 95% of information, k.

In [284…

```
new_img = np.reshape(iml,(87,65))
plt.imshow(new_img, cmap=cm.gray)
plt.title("Reconstruction of first person")
plt.show()
```




### iii. Visualization of the reconstruction of the first person using

In [285…

```
new_img2 = np.reshape(iml2,(87,65))
plt.imshow(new_img2, cmap=cm.gray)
plt.title("95% Reconstruction of first person")
plt.show()
```