SANS

Homework 2: Singular Value Decomposition (SVD)

Oriol Martínez & Imanol Rojas

Introduction

In this report we are going to analyze a code in python that uses Singular Value Decomposition to decompose a high dimensional data matrix. This matrix is a dataset called allfaces.mat which is a library of images of human faces. The dataset contents are the following:

Dataset (allfaces.mat):

- 64 different images (64 different lighting configurations) of 38 different people.
- Each column of allfaces.mat is one of these 64 different images of 38 different people.
- Each image has a resolution of 32256 pixels.
- In total there are 2410 images in the dataset.

Now that the contents of the dataset are clear, we are going to proceed to the explanation of how the code uses the SVD and how the images generated during the execution are obtained.

Code analysis

First, all libraries are included, and the size of the images and font is defined. Then we have the following:

```
mat_contents = scipy.io.loadmat('allFaces.mat')
faces = mat_contents['faces']
m = int(mat_contents['m'])
n = int(mat_contents['n'])
nfaces = np.ndarray.flatten(mat_contents['nfaces'])

allPersons = np.zeros((n*6,m*6))
count = 0

for j in range(6):
    for k in range(6):
        allPersons[j*n : (j+1)*n, k*m : (k+1)*m] =
np.reshape(faces[:,np.sum(nfaces[:count])],(m,n)).T
        count += 1

img = plt.imshow(allPersons)
img.set_cmap('gray')
plt.axis('off')
plt.show()
```

In this part of the code a matrix (allPersons) is created. The contents of this matrix are the 36 first persons. Then the whole image of the matrix is plotted, this is the result:

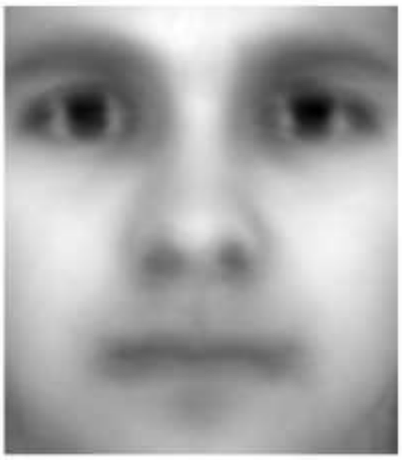
The next step is setting the training set which will be the 64 different lighting images of the first 36 persons. Also, the average column vector of the trainingFaces matrix is calculated (avgFace).

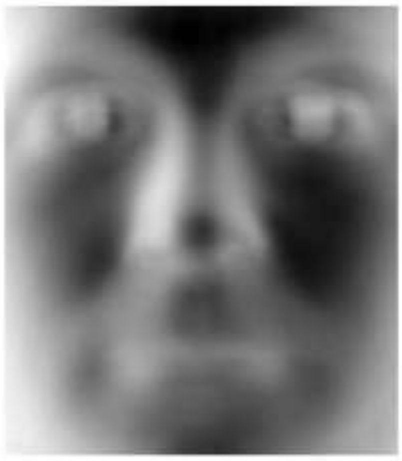
```
# We use the first 36 people for training data
trainingFaces = faces[:,:np.sum(nfaces[:36])]
avgFace = np.mean(trainingFaces,axis=1) # size n*m by 1
```

The next step is creating the eigenface basis that can represent all faces using the training set. This is done as follows:

```
# Compute eigenfaces on mean-subtracted training data
X = trainingFaces - np.tile(avgFace,(trainingFaces.shape[1],1)).T
U, S, VT = np.linalg.svd(X,full_matrices=0)

fig1 = plt.figure()
ax1 = fig1.add_subplot(121)
img_avg = ax1.imshow(np.reshape(avgFace,(m,n)).T)
img_avg.set_cmap('gray')
plt.axis('off')

ax2 = fig1.add_subplot(122)
img_u1 = ax2.imshow(np.reshape(U[:,0],(m,n)).T)
img_u1.set_cmap('gray')
plt.axis('off')
plt.show()
```

The average column vector is subtracted from all faces of the training set and is called matrix X. The SVD of the X matrix is done in the next line of code. We us e the economy SVD to avoid having full matrices and just work with the non-zero singular values. We will end up having the eigenfaces of the mean-subtracted training set. The next plot shows two faces. On the left we see the mean face. This is the average face of all the faces (columns) of the training set. Then, on the left we see the first eigenface of the matrix U. The matrix U has 2410 columns, each column is an eigenface. Every face is a linear combination of the columns of the matrix U.





Now we are going to take the first face of person 37 and we are going to project it on the eigenfaces space of U varying the r number of columns that we take. Then the face will be reconstructed, and we will see how the results change with respect to r.

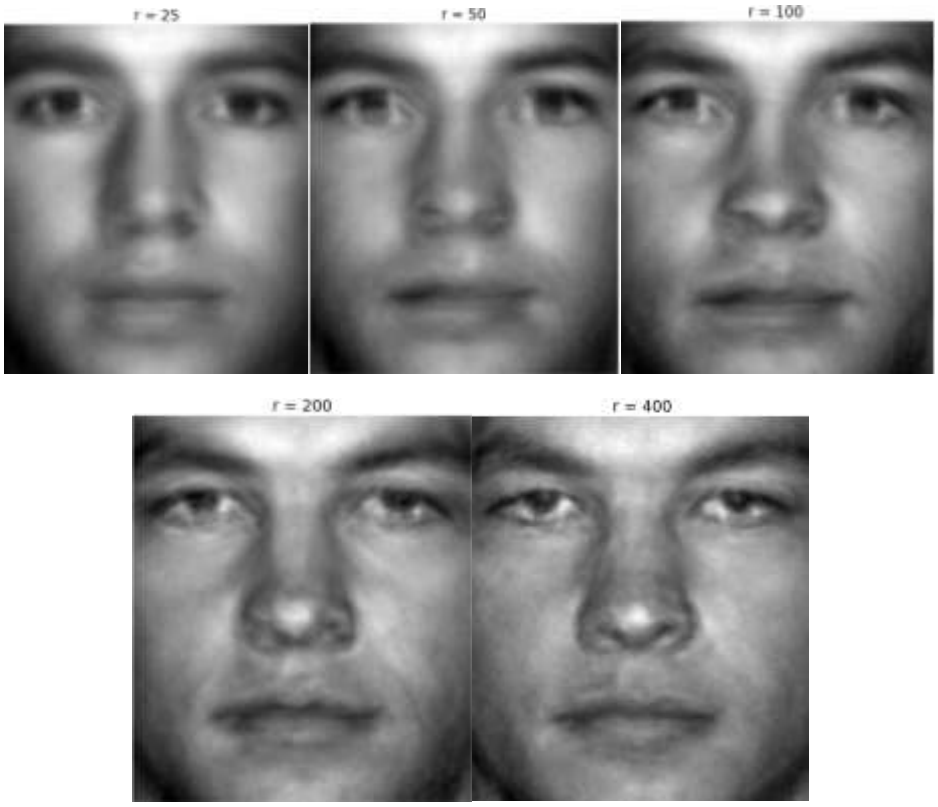
Firs the original 37th image is saved as testFace and is plotted for comparison.

```
testFace = faces[:,np.sum(nfaces[:36])] # First face of person 37
plt.imshow(np.reshape(testFace,(m,n)).T)
plt.set_cmap('gray')
plt.title('Original Image')
plt.axis('off')
plt.show()
```

Then the mean is subtracted from testFace and this is saved to testFaceMS (mean-subtracted). Then the list of different amounts of columns of U is made. Finally, the projection and reconstruction of the image is done all at once and the results are plotted. The formula applied is the following: $\hat{x} = avgFace + U_r U_r^T x_{37MS}$.

```
testFaceMS = testFace - avgFace
r_list = [25, 50, 100, 200, 400, 800, 1600] #1600 takes a lot of CPU time

for r in r_list:
    reconFace = avgFace + U[:,:r]  @ U[:,:r].T @ testFaceMS
    img = plt.imshow(np.reshape(reconFace,(m,n)).T)
    img.set_cmap('gray')
    plt.title('r = ' + str(r))
    plt.axis('off')
    plt.show()
```

The plots for the different values of r (i.e., the number of eigen-modes or columns that we take from U) are the following:

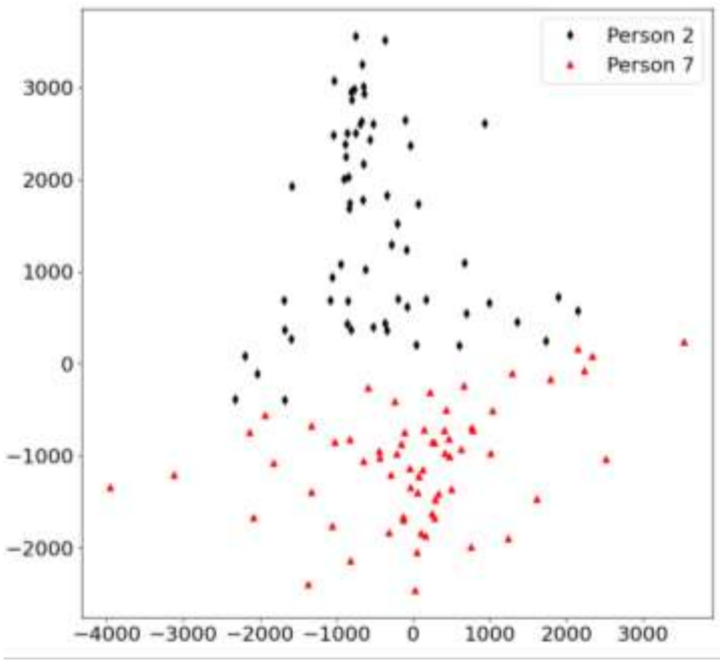


As we can see, as we increase the number of eigenmodes or columns of U that we use to compute the projection of the 37nth face and to reconstruct it, the resulting reconstruction is closer to the original. This is because we are using more and more characteristic traits of the eigenfaces basis.

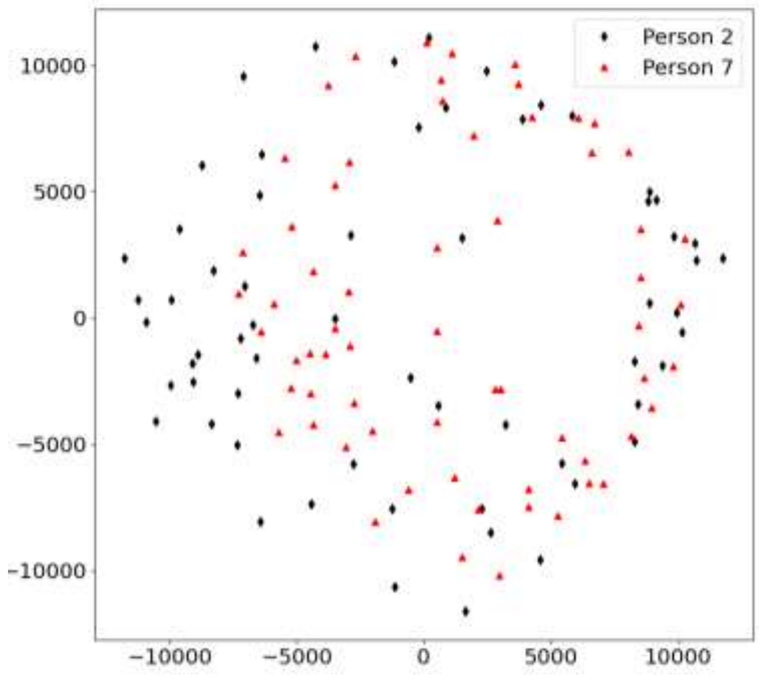
Finally, we are going to see that the SVD is also useful to do image classification. First, persons 2 and 7 are randomly selected and projected into the 5th and 6th columns of U. So, we will get two vectors of dimension 2 (one for person 2 and other for person 7). Then those vectors are plotted in a scatter plot which has PC5 and PC6 (5th and 6th U columns respectively) as its axis.

```
P1num = 2 # Person number 2
P2num = 7 # Person number 7

P1 = faces[:,np.sum(nfaces[:(P1num-1)]):np.sum(nfaces[:P1num])]
P2 = faces[:,np.sum(nfaces[:(P2num-1)]):np.sum(nfaces[:P2num])]

P1 = P1 - np.tile(avgFace,(P1.shape[1],1)).T
P2 = P2 - np.tile(avgFace,(P2.shape[1],1)).T

PCAmodes = [5, 6] # Project onto PCA modes 5 and 6
PCACoordsP1 = U[:,PCAmodes-np.ones_like(PCAmodes)].T @ P1
PCACoordsP2 = U[:,PCAmodes-np.ones_like(PCAmodes)].T @ P2

plt.plot(PCACoordsP1[0,:],PCACoordsP1[1,:],'d',color='k',label='Person
2')
plt.plot(PCACoordsP2[0,:],PCACoordsP2[1,:],'^',color='r',label='Person
7')

plt.legend()
plt.show()
```

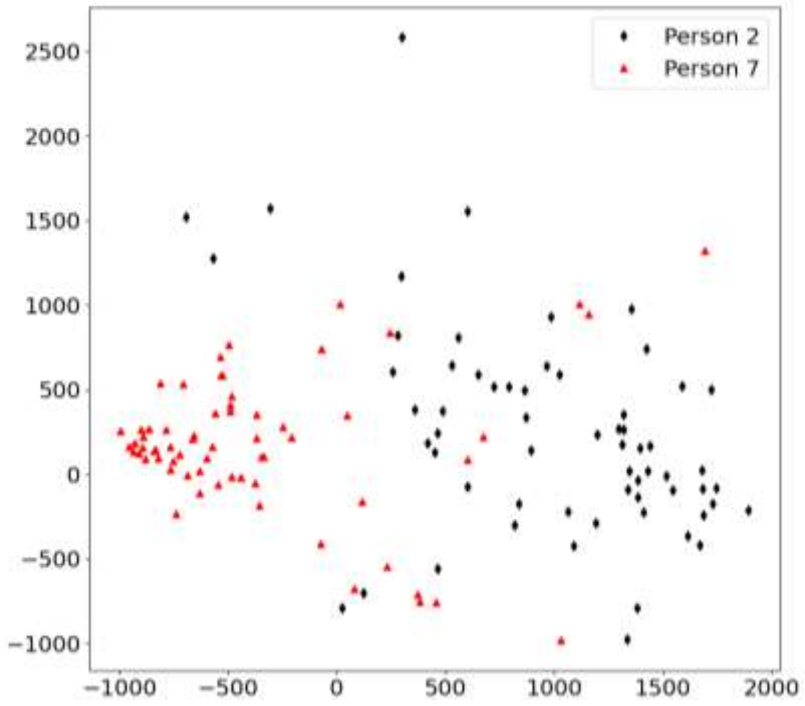
The result of the plotting is the following:



As we see, both persons are clustered and can be differentiated one from the other. The modes selected are traits that can differentiate people. If we took the 1st and the 2nd column of U, we could have similarities between the two persons that we are picking since first columns (first three columns) of U usually encode the most common physical traits of all the training set. If we take the first and the second columns of U, we get a result without clusters like the following graph. This is because all faces have these modes with the same presence.



If we take the 9th and 10th columns of U the resulting graph shows that we start losing the clustering. If we took columns further from this point, we would lose all the clustering as in the previous graph. This is because the information contained in the columns of U has less importance as we go further.


Person 2
Person 7