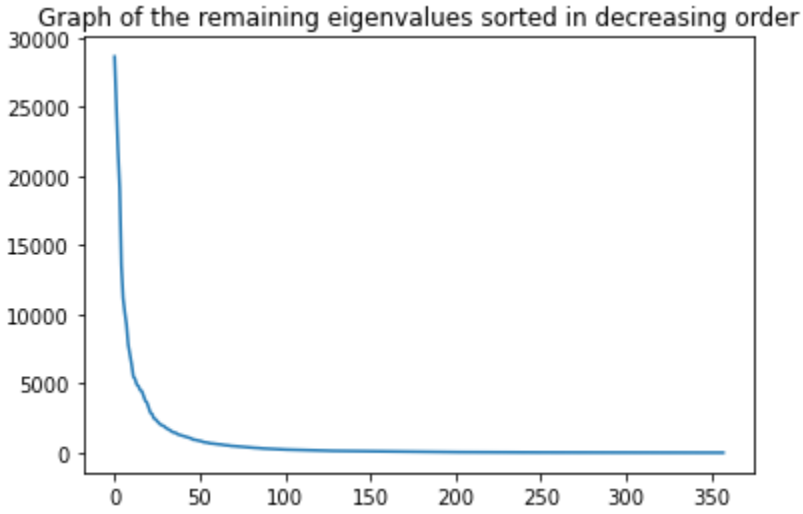
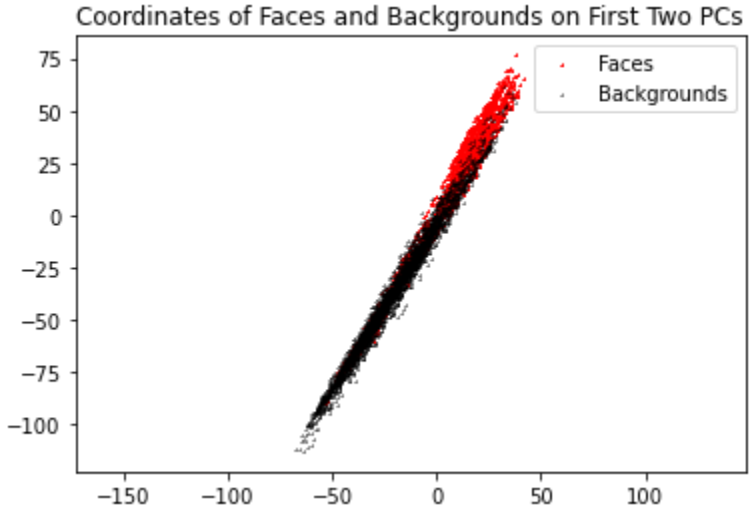
Abelardo Riojas
STA 5635 Spring 2022
Prof. Barbu

Homework 11 Report

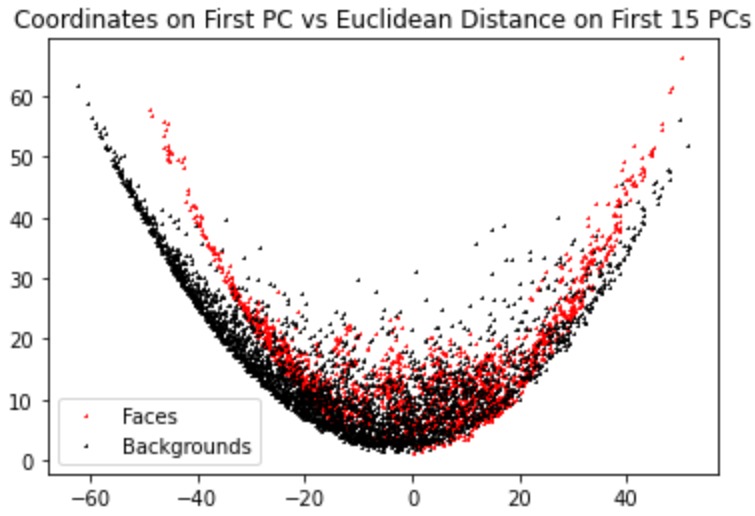
a)



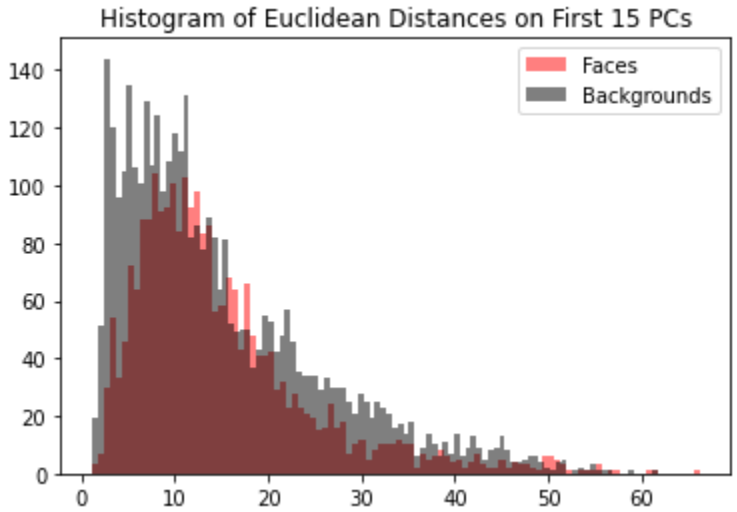
b)



c)



d)



Code below:

```
import cv2
import zipfile
import numpy as np

faces = {}
with zipfile.ZipFile("faces.zip") as facezip:
    for filename in facezip.namelist():
        if not filename.endswith(".pgm"):
            continue # not a face picture
        with facezip.open(filename) as image:
            # If we extracted files from zip, we can use cv2.imread(filename)
instead
            faces[filename] = cv2.imdecode(np.frombuffer(image.read(),
np.uint8), cv2.IMREAD_GRAYSCALE)
```

```
backgrounds = {}
with zipfile.ZipFile("background.zip") as facezip:
    for filename in facezip.namelist():
        if not filename.endswith(".pgm"):
            continue # not a face picture
        with facezip.open(filename) as image:
            # If we extracted files from zip, we can use cv2.imread(filename)
instead
            backgrounds[filename] = cv2.imdecode(np.frombuffer(image.read(),
np.uint8), cv2.IMREAD_GRAYSCALE)
import matplotlib.pyplot as plt

fig, axes = plt.subplots(4,4,sharex=True,sharey=True,figsize=(8,10))
faceimages = list(faces.values())[-16:] # take last 16 images
for i in range(16):
    axes[i%4][i//4].imshow(faceimages[i], cmap="gray")
plt.show()
faceshape = list(faces.values())[0].shape
print("Face image shape:", faceshape)
from sklearn.decomposition import PCA

pca = PCA().fit(facematrix)
eigenfaces = pca.components_ #PCs
eigenvalues = pca.explained_variance_ #Eigenvalues
#discard the first three eigenvalues
eigenvalues = eigenvalues[3:]
plt.plot(eigenvalues)
plt.title('Graph of the remaining eigenvalues sorted in decreasing order')

pca = PCA(n_components=2)
pca.fit(facematrix)
facematrix_pca = pca.transform(facematrix)
print("original shape:   ", facematrix.shape)
print("transformed shape:", facematrix_pca.shape)
facematrix_new = pca.inverse_transform(facematrix_pca) - pca.mean_
plt.scatter(facematrix_new[:, 0], facematrix_new[:, 1], alpha=0.8,
marker='+',s=1, c='red')
plt.axis('equal');
bgmatrix = []
for key,val in backgrounds.items():
    bgmatrix.append(val.flatten())
```

```
# Create facematrix as (n_samples,n_pixels) matrix
bgmatrix = np.array(bgmatrix)
bgmatrix_pca = pca.transform(bgmatrix)
print("original shape:   ", bgmatrix.shape)
print("transformed shape:", bgmatrix_pca.shape)
bgmatrix_new = pca.inverse_transform(bgmatrix_pca) - pca.mean_
plt.scatter(facematrix_new[:, 0], facematrix_new[:, 1], alpha=1,
marker='+',s=1, c='r')
plt.scatter(bgmatrix_new[:, 0], bgmatrix_new[:, 1], alpha=0.5, marker='+',s=1,
c='k')
plt.axis('equal');
plt.title('Coordinates of Faces and Backgrounds on First Two PCs')
plt.legend(['Faces','Backgrounds'])
plt.show()
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Define the PCA object
pca = PCA()

# Run PCA on scaled data and obtain the scores array
T = pca.fit_transform(facematrix)

# Score plot of the first 2 PC
fig = plt.figure(figsize=(8,6))
with plt.style.context(('ggplot')):
    plt.scatter(T[:, 0], T[:, 1], edgecolors='k', cmap='jet')
    plt.xlabel('PC1')
    plt.ylabel('PC2')
    plt.title('Score Plot')
plt.show()


# Compute the euclidean distance using the first 15 PC
euclidean = np.zeros(facematrix.shape[0])
for i in range(15):
    euclidean += (T[:,i] - np.mean(T[:,:15]))**2/np.var(T[:,:15])

colors = [plt.cm.jet(float(i)/max(euclidean)) for i in euclidean]
fig = plt.figure(figsize=(8,6))
with plt.style.context(('ggplot')):
    plt.scatter(T[:, 0], T[:, 1], c=colors, edgecolors='k', s=60)
    plt.xlabel('PC1')
```

```
    plt.ylabel('PC2')
    plt.title('Score Plot')
plt.show()
T_bg = pca.transform(bgmatrix)
# Compute the euclidean distance using the first 15 PC
euclidean_bg = np.zeros(bgmatrix.shape[0])
for i in range(15):
    euclidean_bg += (T_bg[:,i] - np.mean(T_bg[:,:15]))**2/np.var(T_bg[:,:15])
pca = PCA(n_components=1)
pca.fit(facematrix)
facematrix_pca = pca.transform(facematrix)
print("original shape:   ", facematrix.shape)
print("transformed shape:", facematrix_pca.shape)
facematrix_new = pca.inverse_transform(facematrix_pca) - pca.mean_

bgmatrix_pca = pca.transform(bgmatrix)
print("original shape:   ", bgmatrix.shape)
print("transformed shape:", bgmatrix_pca.shape)
bgmatrix_new = pca.inverse_transform(bgmatrix_pca) - pca.mean_
plt.scatter(facematrix_new[:, 0], euclidean, marker='+',s=1, c='r')
plt.scatter(bgmatrix_new[:, 0], euclidean_bg, marker='+',s=1, c='k')
plt.legend(['Faces', 'Backgrounds'])
plt.title('Coordinates on First PC vs Euclidean Distance on First 15 PCs')
plt.show()
plt.hist(euclidean, bins=100, alpha=.5,color='r')
plt.hist(euclidean_bg, bins=100, alpha=.5,color='k')
plt.legend(['Faces', 'Backgrounds'])
plt.title('Histogram of Euclidean Distances on First 15 PCs')
plt.show()
```