Question 4

February 11, 2022

[18]:
```python
#importing necessary packaages
import numpy as np
import PIL
from PIL import Image
import os
import ntpath
import math
import matplotlib.pyplot as plt
```

[19]:
```python
#Placing the input images into numpy array
all_data = []
happy_data = []
sad_data = []
N = 0
directory = r"C:\Users\ujjaw\Desktop\MLSP_Assignments\Ass1\4. Fischer␣
 ↪Faces\Data\emotion_classification\train"
for filename in os.scandir(directory):
    if filename.is_file():
        filename2 = directory + "\\" + ntpath.basename(filename)
        img = Image.open(filename2).resize((100,100))
        np_img = np.array(img)
        flat_array = np.transpose(np.ravel(np_img))
        x = ntpath.basename(filename).split(".")
        if x[1] == "happy":
            happy_data.append(list(flat_array))
        else:
            sad_data.append(list(flat_array))
        N += 1
happy_data = np.transpose(np.array(happy_data))
sad_data = np.transpose(np.array(sad_data))
all_data = np.concatenate((happy_data,sad_data),axis = 1)

# print(all_data.shape)
# print(happy_data.shape)
# print(sad_data.shape)
```

1 PCA Train Data

[20]:
```
#Finding the mean of train data
mean_array = all_data.mean(axis=1)
mean_array = np.reshape(mean_array,(10000,1))
# print(mean_array.shape)
```

[21]:
```
# Without High Dimensional PCA
# Sx = (np.cov(np.transpose(all_data)))

#With High Dimesional PCA
X = all_data - mean_array
Sx = np.matmul(np.transpose(X),X)
Sx = np.multiply(Sx,1/N)
# print(Sx.shape)
```

[22]:
```
#Finding the eigen values and eigen vector of Sx
e_val, e_vec = np.linalg.eig(Sx)

#Sorting Eigen Values and Corresponding Eigen Vectors
idx = e_val.argsort()[::-1]
e_val = e_val[idx]
e_vec = e_vec[:,idx]

# print(e_vec.shape)
# print(e_val)
```

[23]:
```
#Converting Vi to Ui for the first k eigen vector
k = 16
V = e_vec[:,:k]      #Taking first k EVector
V = np.transpose(V)
U = []
for i in range(k):
    temp = np.matmul(all_data, V[i])
    U.append(np.multiply(temp,1/math.sqrt(e_val[i]*20)))

U = np.transpose(np.array(U))
# print(U.shape)
```

[24]:
```
#Reducing the dimensions
reduced_all_data = np.matmul(np.transpose(U), all_data)
# print(reduced_all_data.shape)
reduced_happy_data = np.matmul(np.transpose(U), happy_data)
# print(reduced_happy_data.shape)
reduced_sad_data = np.matmul(np.transpose(U), sad_data)
# print(reduced_sad_data.shape)
```

2 LDA on Train Data

[25]:
```
#Finding mean of different data classes
mean_sad = reduced_sad_data.mean(axis=1)
mean_happy = reduced_happy_data.mean(axis=1)
```

[26]:
```
#Finding mean_diff and Sb
mean_diff = mean_sad-mean_happy
Sb = np.matmul(mean_diff.reshape(k,1),mean_diff.reshape(1,k))
```

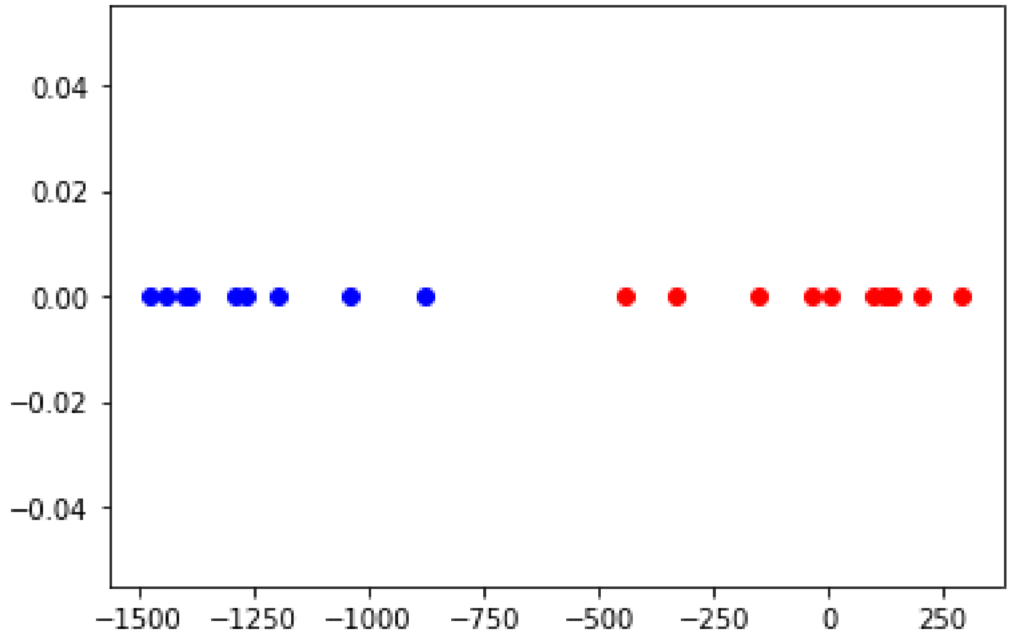
[27]:
```
#Calculating Sw
mean_happy = np.reshape(mean_happy,(k,1))
mean_sad = np.reshape(mean_sad,(k,1))

r1,c1 = reduced_happy_data.shape
r2,c2 = reduced_sad_data.shape
Sw_term1 = np.matmul((reduced_happy_data - mean_happy), np.
 ↪transpose(reduced_happy_data - mean_happy))
Sw_term2 = np.matmul((reduced_sad_data - mean_sad),  np.
 ↪transpose(reduced_sad_data - mean_sad))
Sw = np.multiply(Sw_term1,1/c1) + np.multiply(Sw_term2,1/c2)
```

[28]:
```
Sw_inverse = np.linalg.inv(Sw)
```

[29]:
```
lda_e_val, lda_e_vec = np.linalg.eig(np.matmul(Sw_inverse,Sb))
# print(lda_e_val)

#Sorting Eigen Values and Corresponding Eigen Vectors
idx = lda_e_val.argsort()[::-1]
lda_e_val = lda_e_val[idx]
lda_e_vec = lda_e_vec[:,idx]

required_e_vec = lda_e_vec[:,0].real

required_e_vec = np.reshape(required_e_vec,(k,1))

final_lda_projection_happy = np.matmul(np.transpose(required_e_vec),␣
 ↪reduced_happy_data)
final_lda_projection_sad = np.matmul(np.transpose(required_e_vec),␣
 ↪reduced_sad_data)
# print(final_lda_projection_happy)
# print(final_lda_projection_sad)
```

[30]:
```
plt.scatter(final_lda_projection_happy ,np.zeros((c1,), dtype = int),color =␣
 ↪'b')
plt.scatter(final_lda_projection_sad ,np.zeros((c2,),dtype = int), color = 'r')
```

```
plt.show()
```



3 Applying Dimensionality Reduction and Testing Test Data

[31]:
```
happy_data_test = []
happy_data_test_filename = []
sad_data_test = []
sad_data_test_filename = []
N_test = 0
directory = r"C:\Users\ujjaw\Desktop\MLSP_Assignments\Ass1\4. Fischer␣
 ↪Faces\Data\emotion_classification\test"
for filename in os.scandir(directory):
    if filename.is_file():
        filename2 = directory + "\\" + ntpath.basename(filename)
        img = Image.open(filename2).resize((100,100))
        np_img = np.array(img)/1
        flat_array = np.transpose(np.ravel(np_img))
        x = ntpath.basename(filename).split(".")
        if x[1] == "happy":
            happy_data_test.append(list(flat_array))
            happy_data_test_filename.append(ntpath.basename(filename))
        else:
            sad_data_test.append(list(flat_array))
            sad_data_test_filename.append(ntpath.basename(filename))
```

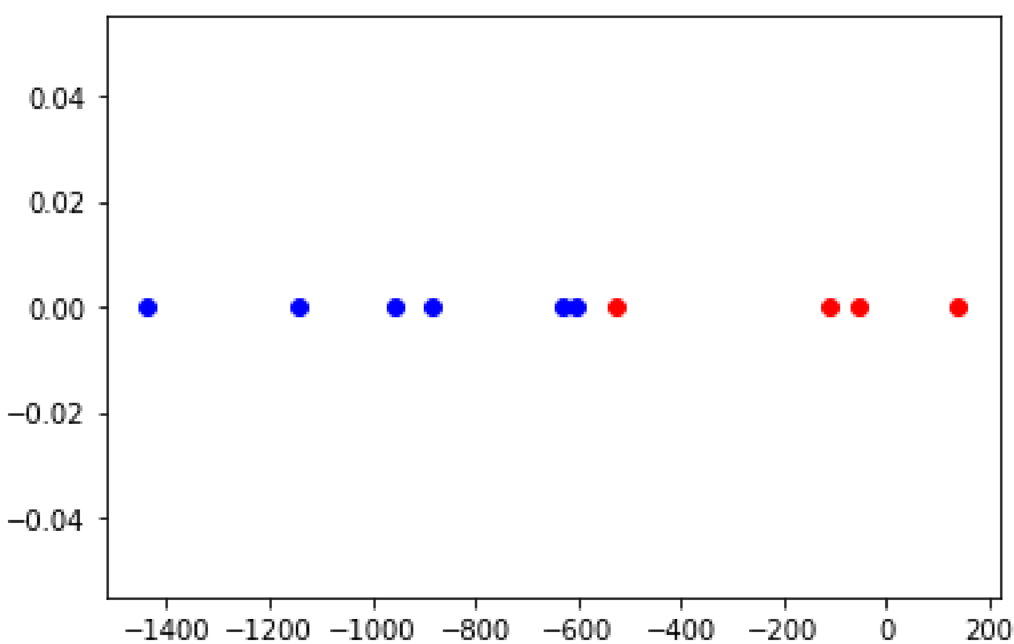
```
        N_test += 1
happy_data_test = np.transpose(np.array(happy_data_test))
sad_data_test = np.transpose(np.array(sad_data_test))
```

[32]:
```
# print(happy_data_test.shape)
# print(sad_data_test.shape)
r1, c1 = happy_data_test.shape
r2, c2 = sad_data_test.shape
```

[33]:
```
reduced_happy_data_test = np.matmul(np.transpose(U), happy_data_test)
# print(reduced_happy_data_test.shape)
reduced_sad_data_test = np.matmul(np.transpose(U), sad_data_test)
# print(reduced_sad_data_test.shape)
```

[34]:
```
final_lda_projection_happy_test = np.matmul(np.transpose(required_e_vec),␣
 ↪reduced_happy_data_test)
final_lda_projection_sad_test = np.matmul(np.transpose(required_e_vec),␣
 ↪reduced_sad_data_test)
# print(final_lda_projection_happy_test)
# print(final_lda_projection_sad_test)
```

[35]:
```
plt.scatter(final_lda_projection_happy_test ,np.zeros((c1,), dtype = int),color␣
 ↪= 'b')
plt.scatter(final_lda_projection_sad_test ,np.zeros((c2,),dtype = int), color =␣
 ↪'r')

plt.show()
```



[36]:
```
total = N_test
correct = 0
for i in range(c1):
    if (final_lda_projection_happy_test[0][i]<-550):
        print(happy_data_test_filename[i],"->","happy")
        correct += 1
    else:
        print(happy_data_test_filename[i],"->","sad")
for i in range(c2):
    if (final_lda_projection_sad_test[0][i]<-550):
        print(sad_data_test_filename[i],"->","happy")
    else:
        print(sad_data_test_filename[i],"->","sad")
        correct += 1

print ("Accuracy =", correct/total * 100,"%")
```

```
subject03.happy.gif -> happy
subject05.happy.gif -> happy
subject08.happy.gif -> happy
subject11.happy.gif -> happy
subject14.happy.gif -> happy
subject15.happy.gif -> happy
subject01.sad.gif -> sad
subject08.sad.gif -> sad
subject14.sad.gif -> sad
subject15.sad.gif -> sad
Accuracy = 100.0 %
```

I have tested the accuracy for different values of k. Accuracy of 100% is observed for k>=16. For k <= 15, the two classes are not completely seperated after projecting in one dimensional LDA.