Pattern Recognition

Assignment 1 report

Team members:

- Karim M. Elsayad, ID: 6023
- Mohamed Mahmoud Mousa, ID: 5375
- Mohamed Wael, ID: 5788
- Amr Essam, ID: 5834

1. Understanding The dataset

The dataset contains a set of faces taken between April 1992 and April 1994 at the Olivetti Research Laboratory in Cambridge, UK.

There are 10 different images of 40 distinct subjects. The files are in PGM format. The size of each image is 92x112, 8-bit grey levels.

2. Generating the data

The images in the dataset are converted into 1-d integer arrays. The numpy method `asarray()` is particularly useful in this step. The final result is a 400x10304 array containing all image data. A label vector is generated where the subject id is the label.

```
# reading the images, flattening them, and creating data and lavel vectors
NUM_SUBJECTS = 40
NUM_IMAGE_PER_SUBJECT = 10
data_matrix = np.zeros((400, 10304))
image_index = 0
label_vector = np.zeros(400)
for i in range(1, NUM_SUBJECTS + 1):
  for j in range(1, NUM_IMAGE_PER_SUBJECT + 1):
    image = PImage.open(f"dataset/s{i}/{j}.pgm")
    data = np.asarray(image)
    data_matrix[image_index, :] = data.flatten()
    label_vector[image_index] = i
    image_index += 1
```

Figure 1: Code to generate the data

3. Splitting the data

The initial data matrix was split in half, where even rows where for training and odd rows where for testing. The label vectors were split accordingly.

```
# odd rows are training data
image_index = 0
for i in range(1, 400, 2):
    training_data[image_index, :] = data_matrix[i]
    training_labels[image_index] = label_vector[i]
    image_index += 1
```

```
# even rows are testing data
image_index = 0
for i in range(0, 400, 2):
    testing_data[image_index, :] = data_matrix[i]
    testing_labels[image_index] = label_vector[i]
    image_index += 1
```

Figure 2: splitting the data matrix into training and test sets

4. Performing classification using PCA

After determining principal components and projecting both training and testing data, classification analysis was performed using K-Nearest-Neighbors. Various thresholds were tested [0.8, 0.85, 0.9, 0.95], at different values of K [1, 3, 5, 7].

Observations:

It seems that as the value of alpha increases, the accuracy tends to be higher for various values of K, however, for one value of alpha, the accuracy tends to decrease the more neighbors we consider.

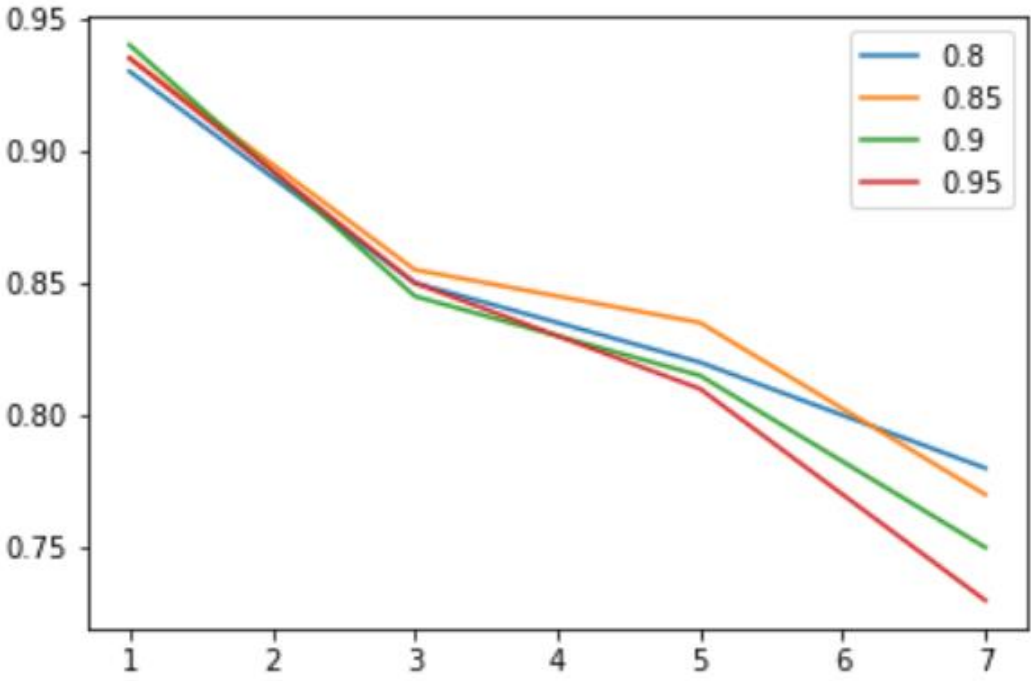

Figure 3: Plot of number of neighbors considered to accuracy in case of PCA

5. Performing Classification using LDA

Unlike with PCA, the classification with LDA was performed on one case, in which we consider 39 dominant eignevectors.

Observations:

LDA reports higher accuracy for lower values of K compared to PCA, but for higher values of K, the accuracy is much lower

```
plt.plot(K_neighbours, accuracy_list)
```

```
Out[428]: [<matplotlib.lines.Line2D at 0x27238ab30b8>]
```

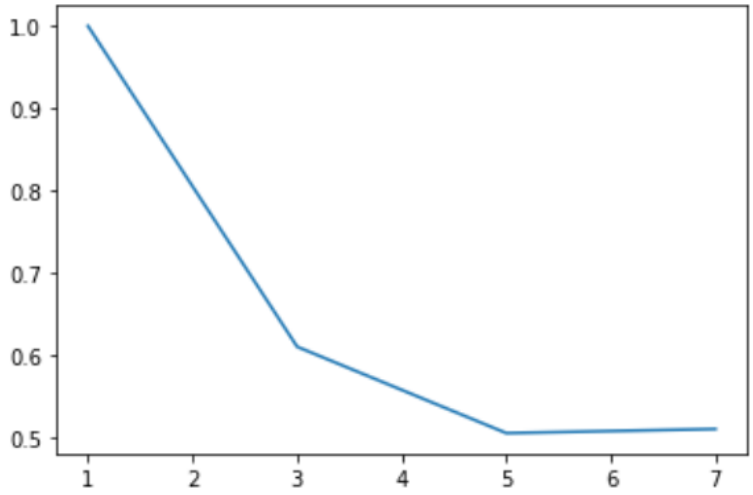

Figure 4: Plot of number of neighbors considered to accuracy in case of LDA

6. faces vs. Non-faces

The classification problem was performed using 900 image data set containing both the att faces and other image data containing various other thing, such as planes and fruit.

The data was split in the same way (50:50) as the previous problems. Results were similar to the previous LDA classification problem.

Unfortunately, the team wasn't able to determine how many dominant vectors to use, and the results were suspicious. Accuracy was high for all sets of dominant vectors considered, suggesting a fault in the methodology. The images couldn't be re-produced from their vector representation. This has hindered performing the remainder of the 7th task.

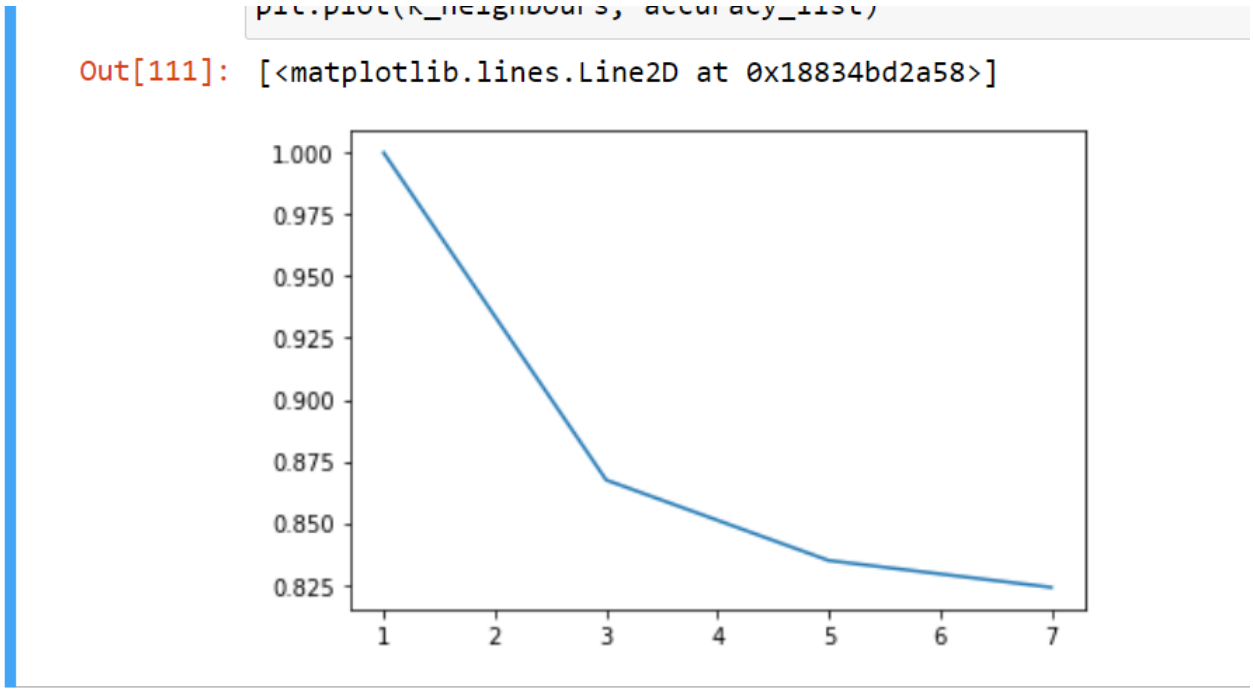

Figure 5: Plot of number of neighbors considered to accuracy in case of 2-class problem