Submission for Programming Assignment 1

CS 427: Mathematics for Data Science, Autumn 2020-21

Group 9: K. Sai Anuroop (170030035), Rohan Savakar (170030033)

November 22, 2020

The code performs Principal Component Analysis on the set of images provided to us from Olivetti faces dataset. It then randomly chooses an image, reconstructs it using 4096 components, x components and x components, where x is provided as an input argument by the user. We interpret x as the minimum number of components required to reconstruct the image such that the face of the person is discernible. Below is an exhaustive list of all the 30 images provided to us along with their reconstructed counterparts as per the scheme mentioned above.

Original	4096	x	< x
		5 35 35 46 40 40 41 40 41 41 41 41 41 41 41 41 41 41 41 41 41	10 27 28 40 40 40 40 40 40 40 40 40 40 40 40 40
0.png	-	3100	353
	0 10 - 22 - 30 - 30 - 30 - 30 - 30 - 30 - 40 - 40 - 40 - 40 - 40 - 40 - 40 - 4	0 23 25 26 26 27 28 28 28 28 28 28 28 28 28 28 28 28 28	1 10 27 28 40 40 40 40 40 40 40 40 40 40 40 40 40
1.png		3250	476
36	0 10 20 20 20 40 40 44	0 130 221 220 220 450 450 450	8 33 00 33 60 33 00
2.png		3300	1220
130	0 10 20 20 20 20 20 20 20 20 20 20 20 20 20	0 30 30 30 30 30 30 30 30 40 40 40 40 40 40 40 40 40 40 40 40 40	6 10 20 20 20 20 20 20 20 20 20 20 20 20 20
3.png	-	3250	476
3		0 13 23 23 23 44 13 44 14	10 2 2 3 3 4 4 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
4.png		3500	817
1		0 29 29 29 29 29 29 29 29 29 29 29 29 29	10 27 28 28 28 28 28 28 28 28 28 28 28 28 28
5.png	-	3650	2990

Original	4096	x	< x
50		0 33 32 33 30 40 38 44	130
6.png	-	3650	2990
66	0 30 2 37 30 40 50 50 51 51 52 53 54 54 54 54 54 54 54 54 54 54 54 54 54	0 30 20 30 40 40 40 40	1 13 13 13 13 13 13 13 13 13 13 13 13 13
7.png		3250	476
	0 30 30 40 50 50 50 50 50 50 50 50 50 50 50 50 50	0 33 32 32 32 43 44 44 45	13 13 14 14 15 15 15 15 15 15 15 15 15 15 15 15 15
8.png		3300	1220
0	0 22 23 24 24 25 24 25 25 25 25 25 25 25 25 25 25 25 25 25	0 30 30 30 30 30 40 40 40 40	5 22 23 23 40 20 10 10
9.png	-	3350	3105
		0 10 20 20 20 20 20 20 20 20 20 20 20 20 20	10 27 28 28 40 40 40 40 40 40 40 40 40 40 40 40 40
10.png		3000	1331
11.png	0 32 32 33 65 60 120 20 60 36 64	30 30 40 40 40 40 40 40 40 40 40 40 40 40 40	2990
	0	0	'KILLEYNOU AND I
	20 20 20 20 20 20 20 20 20 20 20 20 20 2	20 27 28 28 28 28 28 28 28 28 28 28 28 28 28	10 2 2 3 4 4 5 5 6 7 7 7 7 8 6 7 8 7 8 7 8 7 8 7 8 7 8 7
12.png	-	3050	541
13.png		31 32 33 34 34 35 35 35 35 35 35 35 35 35 35 35 35 35	2098
	0		
		20 22 23 24 24 25 25 26 26 26 26 26 26 26 26 26 26 26 26 26	10 20 20 20 20 20 20 20 20 20 20 20 20 20
14.png	-	3200	2098

Original	4096	x	< x
	10 17 17 17 18 18 18 18 18 18 18 18 18 18 18 18 18	6 23 27 27 28 24 25 25 25 25 25 25 25 25 25 25 25 25 25	
15.png	-	3100	353
	0 20 20 20 40 50 50 50 50 50 50 50 50 50 50 50 50 50	6 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	1 10 20 20 20 40 30 40 50
16.png	<u>-</u>	3000	1331
	0 20 20 30 40 0 10 20 80 40 40 40	0 22 22 23 24 24 25 25 25 25 25 25 25 25 25 25 25 25 25	1 20 20 20 20 20 20 20 20 20 20 20 20 20
17.png	<u>-</u> -	3200	2098
	30 29 30 40 30 45	0 10 22 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	30 20 30 40 30 40 40 40
18.png	-	3100	353
		6 37 37 38 38 38 38 38 38 38 38 38 38 38 38 38	
19.png		3050	541
100	10 10 10 10 10 10 10 10 10 10 10 10 10 1	10 20 20 20 40 10 40 40	30 20 20 20 20 20 20 20 20 20 20 20 20 20
20.png		3000	1331
100		6 22 22 23 24 24 25 25 25 25 25 25 25 25 25 25 25 25 25	
21.png	-	2700	1587
1	10 20 20 20 20 20 20 20 20 20 20 20 20 20		
22.png	-	2700	1587

Original	4096	x	< x
63)	0 33 2 22 30 40 48 48	6 22 22 23 24 24 25 25 25 25 25 25 25 25 25 25 25 25 25	1 10 22 22 23 24 24 25 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 25 26 2
23.png	-	3000	1331
3	0 30 - 22 - 23 - 24 - 24 - 24 - 24 - 24 - 24	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
24.png	<u>-</u>	2700	1587
63	0 20 20 30 40 50 50 50 50	0 30 32 33 46 46 50 46 47 48 48 48 48 48 48 48 48 48 48 48 48 48	1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
25.png		2750	523
0 0		5 25 25 25 25 25 25 25 25 25 25 25 25 25	
26.png	-	2900	1744
27.png		2900	10 20 30 40 40 10 10 1744
2117118	•	•	
200	30 - 32 - 33 - 40 - 46 - 46	20 22 22 23 24 24 24 24 24 24 24 24 24 24 24 24 24	20 20 20 20 20 20 40 40 40 40
28.png	<u>-</u>	2850	2450
63)	0 19 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	0 10 27 27 28 20 40 30 40	6 10 20 20 20 40 40 40 40 40
29.png	-	2700	1587

is the average of all the values of x listed in the above table, which we consider to be the value of x that is required to discern the faces, according to our discretion.

```
1 """
2 This code performs PCA on the set of images provided in the format <number>.png,
3 reconstructs a randomly chosen image from the input set using [4096, k, <k] Principal
      Components,
4 and saves in ./output/img<number> directory.
5 Note that this code has to be placed in the same directory as the set of input images (or else
       change the path in the code accordingly)
7 import os
8 import sys
9 import numpy as np
10 import pandas as pd
11 import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
14 from sklearn.decomposition import PCA
15 from glob import iglob
16 from numpy import mean
17 from numpy import cov
18 from numpy.linalg import eigh
19 from pathlib import Path
21 # checking if sufficient input arguments are provided and if provided, are correct or not
22 if(len(sys.argv)<2 or int(sys.argv[1])>4096 or int(sys.argv[1])<1):</pre>
      print("Usage: python3 pca3.py <num_principal_components_to_retain>")
      print("Note: 1 <= num_principal_components_to_retain <= 4096")</pre>
24
25
      exit(1)
26
\ensuremath{\mathtt{z}}_{7} # dataframes to store images of a person's face
28 faces_0 = pd.DataFrame([])
29 faces_1 = pd.DataFrame([])
30 faces_2 = pd.DataFrame([])
all_faces = pd.DataFrame([])
32
_{33} # convert 64\!*\!64 images to a pandas dataframe of 4096 size each
34 for path in iglob("*.png"):
      img=mpimg.imread(path)
35
      # print(int(''.join(path.split('.')[0])))
36
      face = pd.Series(img.flatten(),name=path)
37
38
      all_faces = all_faces.append(face)
       if int(int(path[0:path.find('.')])/10)==0:
39
           # print(int(path[0:path.find('.')]))
40
41
          faces_0 = faces_0.append(face)
      elif int(int(path[0:path.find('.')])/10) == 1:
42
           # print(int(path[0:path.find('.')]))
43
          faces_1 = faces_1.append(face)
44
45
      else:
           # print(int(path[0:path.find('.')]))
46
           faces_2 = faces_2.append(face)
48
_{49} # uncomment to choose a random person to perform PCA on the set of images of "that" person
50 """
_{\rm 51} # choose a random person to perform PCA
52 face_num = random.randint(0,((len(faces_0)+len(faces_1)+len(faces_2))/10)-1)
53 # choose a random image of the person to compare with
s4 which_face = random.randint(0,10)
55 fit_which_face = faces_0
56 if face_num == 0:
57
      fit_which_face = faces_0
      my_face = faces_0[which_face]
58
59 elif face_num == 1:
     fit_which_face = faces_1
60
61
      my_face = faces_1[which_face]
62 else:
     fit_which_face = faces_2
my_face = faces_2[which_face]
66 # sort indices of images
all_faces = all_faces.assign(indexNumber=[int(''.join(i.split('.')[0])) for i in all_faces.
      index])
68 all_faces.sort_values(['indexNumber'], ascending = [True], inplace = True)
69 all_faces.drop('indexNumber', 1, inplace = True)
_{71} temp = all_faces # temp: 30 x 4096
72 temp = temp.values
73 mean_matrix = mean(temp.T, axis=1)
```

```
74 centred_matrix = temp - mean_matrix
75 cov_matrix = cov(centred_matrix.T) # finding covariance matrix
76 values, vectors = eigh(cov_matrix) # finding eigenvectors (Principal Components) of covariance
       matrix
vectors = np.real(vectors) # vectors: 4096 x 4096
79 # sort eigenvectors (Principal Components) based on their eigenvalues in descending order
80 idx = values.argsort()[::-1]
81 values = values[idx]
82 vectors = vectors[:,idx]
retention_vector = [4096, int(sys.argv[1]), random.randint(0,int(sys.argv[1]))] # array
      storing number of eigevectors (Principal Components) to be used while reconstructing image
85 # choose an image to reconstruct and create an output directory for the same
reconstruct_img_num = random.randint(0,29)
87 # reconstruct_img_num = 18
88 filename = "./output/img"+str(reconstruct_img_num)
89 Path(filename).mkdir(parents=True, exist_ok=True)
91 for k in retention_vector: # reconstruct the chosen image using [4096, k, <k] eigevectors (
      Principal Components)
       # retain only k eigevectors (Principal Components)
92
      retained_values = values[:k] # choose this number k
93
       retained_vectors = vectors[:k] # retained_vectors: k x 4096
94
      # Summary of matrix dimensions
96
      # temp: 30 x 4096
97
98
       # retained_vectors: k x 4096
      # projected_matrix: 30 x k
99
       \# recon_matrix: 30 x 4096
100
       # reconstruct images
      projected_matrix = temp.dot(retained_vectors.T) # projected_matrix: 30 x k
       recon_matrix = projected_matrix.dot(retained_vectors)+mean_matrix # recon_matrix: 30 x
104
       4096
       plt.imshow(recon_matrix[reconstruct_img_num].reshape(64,64),cmap="gray")
       plt.savefig(filename+"/"+str(k)+"pc_recon_"+str(reconstruct_img_num)+".png")
106
107
```

Python code to perform PCA on images



Scan this QR code to access the GitHub repository of my homweork solutions at https://github.com/ksanu1998/MDS_HW_Solutions