Jigyas Sharma

Dr. Sumaiya Shomaji

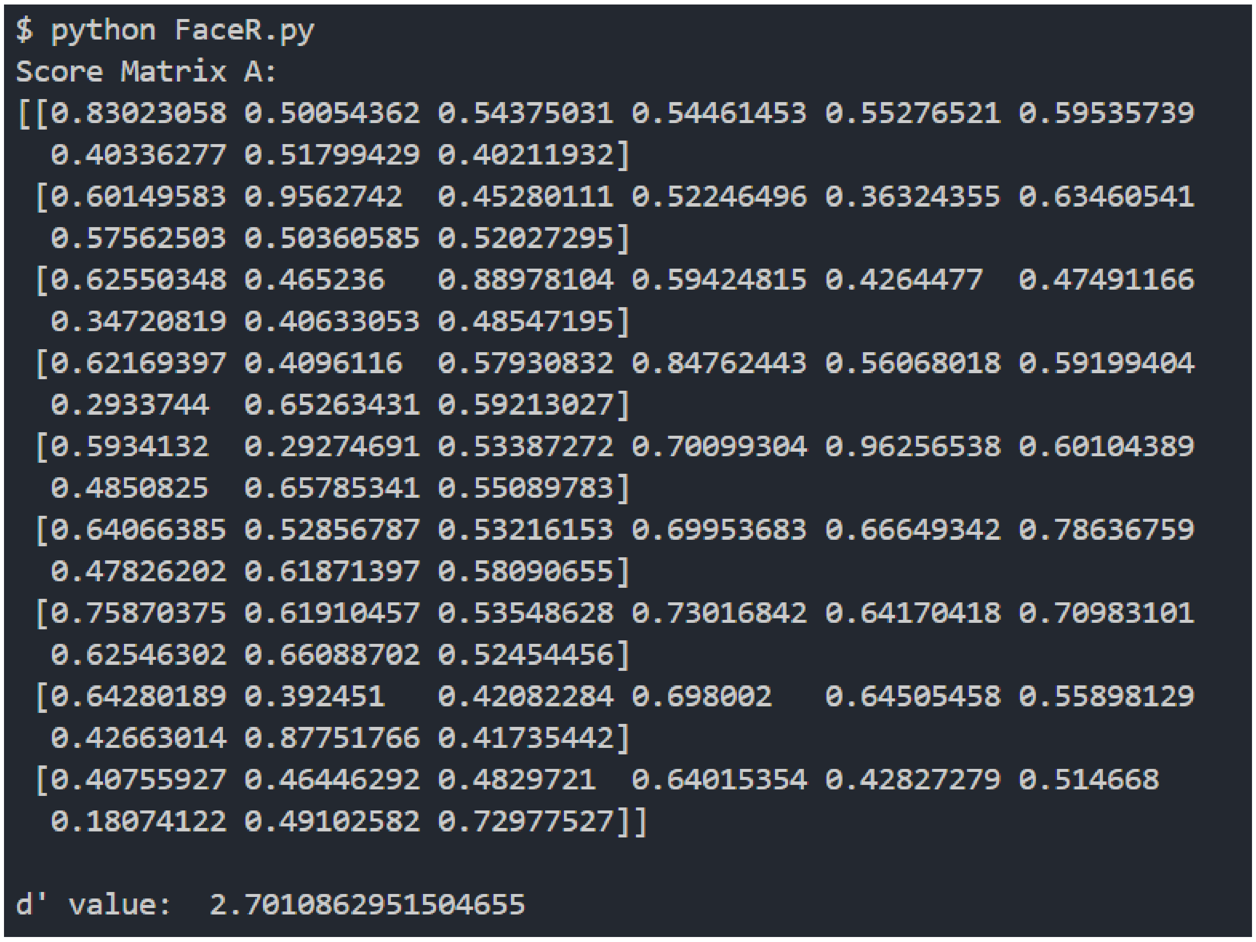
EECS 841

11 April 2024

Homework 4

Section I

1)



```
$ python FaceR.py
Score Matrix A:
[[0.83023058 0.50054362 0.54375031 0.54461453 0.55276521 0.59535739
  0.40336277 0.51799429 0.40211932]
 [0.60149583 0.9562742  0.45280111 0.52246496 0.36324355 0.63460541
  0.57562503 0.50360585 0.52027295]
 [0.62550348 0.465236   0.88978104 0.59424815 0.4264477  0.47491166
  0.34720819 0.40633053 0.48547195]
 [0.62169397 0.4096116  0.57930832 0.84762443 0.56068018 0.59199404
  0.2933744  0.65263431 0.59213027]
 [0.5934132  0.29274691 0.53387272 0.70099304 0.96256538 0.60104389
  0.4850825  0.65785341 0.55089783]
 [0.64066385 0.52856787 0.53216153 0.69953683 0.66649342 0.78636759
  0.47826202 0.61871397 0.58090655]
 [0.75870375 0.61910457 0.53548628 0.73016842 0.64170418 0.70983101
  0.62546302 0.66088702 0.52454456]
 [0.64280189 0.392451   0.42082284 0.698002   0.64505458 0.55898129
  0.42663014 0.87751766 0.41735442]
 [0.40755927 0.46446292 0.4829721  0.64015354 0.42827279 0.514668
  0.18074122 0.49102582 0.72977527]]

d' value:  2.7010862951504655
```

2) In the above image, we can see that the d' value is 2.701086. A higher d’ value signifies a greater difference between the genuine and imposter scores, it also results in fewer false recognitions and overlaps, therefore, it is better to have a higher d' value.

3) If we were consider three channel images, it would mean that there would be more information available for the system to make predictions, and provide more accurate matches, however, the performance of the system might go down as there will be more information for the system to process

Section II(Code)

```
'''
@author: Jigyas Sharma
@organization: University of Kansas
@Course: EECS 841(Computer Vision)
'''
#Library Imports
import cv2
import numpy as np
import os


def getImages(dir_path):
  """
  Load grayscale images from the specified directory path.


  Args:
  - dir_path: Path to the directory containing the images.


  Returns:
  - images: List of grayscale images.
  """
  images = []
  for file in os.listdir(dir_path):
    img = cv2.imread(os.path.join(dir_path, file), cv2.IMREAD_GRAYSCALE)
```

```
        if img is None:
            print(f"Error: Unable to load image {file}")
        else:
            images.append(img)
    return images

def normalizedCorrCoeff(img1: np.ndarray, img2: np.ndarray):
    """
    Calculate the normalized correlation coefficient between two images.

    Args:
    - img1: First image.
    - img2: Second image.

    Returns:
    - r: Normalized correlation coefficient.
    """
    x = img1.reshape((-1, 1))
    y = img2.reshape((-1, 1))
    xn = x - np.mean(x)
    yn = y - np.mean(y)
    r = (np.sum(xn * yn)) / (np.sqrt(np.sum(xn**2)) * np.sqrt(np.sum(yn**2)))
    return r

def calculateScoreMatrix(gallery_set, probe_set):
    """
    Calculate the score matrix for face recognition.

    Args:
    - gallery_set: List of images in the gallery set.
    - probe_set: List of images in the probe set.

    Returns:
    - score_matrix: One-dimensional array containing similarity scores between probe
and gallery images.
    """
    score_matrix = []
    for i in probe_set:
        for j in gallery_set:
            temp = normalizedCorrCoeff(i, j)
            score_matrix.append(temp)
    return score_matrix




def calculate_d_prime(Score_Array):
    """
    Calculate the d' value for face recognition performance evaluation.

    Args:
    - score_matrix: Matrix containing similarity scores between probe and gallery
images.
```

```
    Returns:
    - d_prime: Decidability index value.
    """
    gal_score = []
    prob_score = []
    for i in range(99):
        for j in range(99):
            temp = Score_Array[i, j]
            if i == j:
                gal_score.append(temp)
            else:
                prob_score.append(temp)

    mu1 = np.mean(gal_score)
    mu0 = np.mean(prob_score)
    sigma1 = np.std(gal_score)
    sigma0 = np.std(prob_score)

    d_prime = np.sqrt(2) * np.abs(mu1 - mu0) / np.sqrt(sigma1**2 + sigma0**2)
    return d_prime

# Load images from directories
gallery_set = getImages('./GallerySet')
probe_set = getImages('./ProbeSet')
```

```
#Calculate Score Matrix by calling the function

Score_Matrix = []

Score_Matrix = calculateScoreMatrix(gallery_set, probe_set)

Score_Matrix = np.array(Score_Matrix)

Score_Array = Score_Matrix.reshape(100,100)

#Print the sub 10x10 matrix for validation

print(f'Score Matrix A:\n{Score_Array[0:9, 0:9]}')

#Calculate the d prime value

d_prime = calculate_d_prime(Score_Array)

print("d' value:", d_prime)
```