Team Members: Hunter Glass, James Hanselman, Alice Kuang, Mackenzie Meier, Evan Powell

Dr. Sumaiya Shomaji

EECS 700

7 December 2023

Project: Analyzing and Improving a Basic Face Biometric System

**Part I: (25 points)**

You build the basic face biometric system in this part using the given data “ProbeSet.rar” and “GallerySet.rar” from HW3. Complete the following:

(a) Provide a snippet of the A[0:9,0:9] matrix. (5 points)

[0.7283465824473174 0.4202354359009726 0.3768899369395291 0.36044800133756827 0.4671048878214072 0.46062844517013307 0.5870864521020758 0.4885150568769557 0.7619106584434582 0.24735125877633413 ]

[0.9717296312687385 0.472864247033211 0.36946217544792054 0.369453526109065 0.3579179533969418 0.5979484272323657 0.5911136359266695 0.5293507709167388 0.7156768601475727 0.20032714837948024 ]

[0.48902435437134995 0.8686677657040898 0.46442939102411246 0.5493785751062373 0.631639070504463 0.4087235082966626 0.35773978463705436 0.5707686767754743 0.5990291117943823 0.5093812963971439 ]

[0.5201920604685526 0.9671106784716013 0.3912238765789851 0.560155691844751 0.5826142469302772 0.44390329411244855 0.34601354675879215 0.5406310700360121 0.562017023786024 0.5138363367221811 ]

[0.46810697681718433 0.5213682246246901 0.774228654428116 0.7373969817561983 0.44232899337085735 0.779941632697075 0.35394393616974573 0.7278638414588914 0.3879071668488262 0.6928335757728133 ]

[0.5642658000021203 0.5400248887097048 0.7778152701889427 0.7266450310964511 0.4827199860286597 0.8249327429888864 0.45484551449603605 0.7808721141275984 0.47864828425782774 0.6138874856110181 ]

[0.3885612271153826 0.6381623426075878 0.6303399751420474 0.8771924057460029 0.5940171044883911 0.635567511703692 0.46419136164331315 0.6609154551246055 0.446874763259065 0.7279456203287894 ]

[0.42881486029080673 0.5610230916079554 0.6690702809968345 0.9577345111031002 0.4766945537291173 0.6871833184237259 0.44924316389040264 0.6991224252913706 0.4275906685205047 0.6310674779332868 ]

[0.43435637573978103 0.6768523787602472 0.647974334689092 0.6507617804941148 0.7676369871987878 0.5146355137941633 0.40225637994463687 0.7961899958497967 0.5921612982894154 0.6575148906032003 ]

[0.482187566498275 0.7011790502492923 0.6883101766867211 0.6865759456180086 0.7271015045052918 0.5813186826714405 0.3755596134363526 0.8220056384691206 0.5881450079858028 0.68859885207006 ]

(b) Plot the genuine and impostor score distributions in a reasonably comparable window. (5 points)

A graph showing the face matching score distribution

Description automatically generated

(c) Report the decidability index value. (5 points)

Decidability Index**:** 2.3338949099424617

(d) Plot the Receiver Operating Curve (FAR vs. FRR). (5 points)

A graph with a line

Description automatically generated

(e) What is the EER? (5 points)

A graph with a red line and blue line

Description automatically generated

Through graphical methods, we find that the threshold value is 0.6980853679957842.

Similarly, the Equal Error Rate can be found by finding the intersection of a 45 degree line and the intersection, or by reading the y intersection value in the above graph, which is 9.394695239768136 %.

**Part II.**

Part I represents a very basic face biometric system which considers the raw pixel intensity values as features as well as a normalized correlation coefficient as comparator. As mentioned in the Objective, in Part II you will propose and implement a method for improving/attempting-to-improve the performance of the above basic face biometric system. [Hint: You may consider different feature representations, pre-processing techniques, matching algorithms, etc. or combination of them.]

Now answer the following:

1. Briefly introduce your proposed method. Clearly define the differences between proposed vs. basic face biometric system. Also, discuss the justification of your choice of the proposed method. *(2.5 + 2.5 + 5 = 10 points)*
2. Generate a new score matrix (*A\_new*) for your proposed solution. Your new score matrix *A\_new* will also be a 200x100 matrix where *A\_new[i,j]* denotes the match score generated by comparing the *i*-th probe with the *j*-th gallery data. Provide a snippet of *A\_new[0:9,0:9]. (5 points)*
3. Plot the genuine and impostor score distributions in a reasonably comparable window. Discuss if the distribution is better than that of the basic system. *(5 + 5 = 10 points)*
4. Report the decidability index value. Discuss if the separation is better than that of the basic system. *(5 + 5 = 10 points)*
5. Plot the Receiver Operating Curve (FAR vs. FRR). Discuss if the verification performance is better than that of the basic system. *(5+ 5 = 10 points)*
6. What is the EER of the proposed system? Discuss if the verification performance is better than that of the basic system. *(5+ 5 = 10 points)*
7. The goal of Part II is to revise the basic system in such a way so that it can provide better recognition performance. Quantify the amount of improvement by the metric called “*Improvement Factor (IF)*” considering the given formula. *(20 points)*

*IF = round (delta\_ d’, 2)* where *delta\_d’ = d’ of your proposed system – d’ of basic system*

A questionnaire with black text

Description automatically generated

\*

**General Comment:** For this project, you will receive credits for how much improvement you could earn (if any), thoroughness of your work, justification of implementations, and your analytical ability while analyzing as well as comparing the performance of your proposed system vs. the basic system.

**\***

**Part III (Code).**

# PART I **==========================================================================**

import patoolib

import cv2

import math

import numpy as np

from numpy import \*

import matplotlib.pyplot as plt

import statistics

from shapely.geometry import LineString

import seaborn as sns

sns.set\_theme()

patoolib.extract\_archive("GallerySet.rar", outdir="GallerySet")

patoolib.extract\_archive("ProbeSet.rar", outdir="ProbeSet")

# Given formula from appendix; use as similarity measure.

def normalizedCorrelationCoefficient(img1: np.ndarray, img2: np.ndarray):

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

score\_matrix = np.zeros([200,100])

# record scores to correlation matrix

pi = 2

for i in range(0, 200):

if i%2==0:

pi = 2

else:

pi = 3 # probe idx

img1\_path = "ProbeSet/subject" + str(math.ceil((i+1) / 2)) + "\_img" + str(pi) + ".pgm"

img1 = cv2.imread(img1\_path, cv2.IMREAD\_GRAYSCALE)

for j in range(0, 100):

img2\_path = "GallerySet/subject" + str(j+1) + "\_img1.pgm"

img2 = cv2.imread(img2\_path, cv2.IMREAD\_GRAYSCALE)

score\_matrix[i][j] = normalizedCorrelationCoefficient(img1, img2)

print('10x10 matrix')

for i in range(10):

print('[', end = '')

for j in range(10):

print(score\_matrix[i][j], end = ' ')

print(']')

# filter by idx i = j to get arrays containing two distributions.

def separate\_distributions(arr, genuine, imposter):

for i in range(200):

for j in range(100):

if math.floor(i/2) == j:

genuine.append(arr[i][j])

else:

imposter.append(arr[i][j])

# plot w/ Seaborn distplot

def plotNormalizedDist(genuine, imposter, title):

sns.histplot(data=genuine, bins=35, kde=True, stat='density')

sns.histplot(data=imposter, bins=25, kde=True, stat='density')

plt.xlabel('Threshold')

plt.ylabel('Probability Density')

plt.title(title + ' Normalized Distribution')

# initialize two arrays to hold genuine and imposter scores

genuine=[]

imposter=[]

separate\_distributions(score\_matrix, genuine, imposter)

plotNormalizedDist(genuine, imposter, 'Face Matching Score Distribution')

#(c) Report the decidability index value. (10 points)

def d\_value\_calc(genuine, imposter):

m1 = statistics.mean(genuine)

s1 = statistics.stdev(genuine)

m0 = statistics.mean(imposter)

s0 = statistics.stdev(imposter)

return math.sqrt(2) \* abs(m1 - m0) / math.sqrt(s1\*s1 + s0\*s0)

print('Decidability value: ', d\_value\_calc(genuine, imposter))

#(d) Plot the Receiver Operating Curve (FAR vs. FRR).

# FAR

dimensions = np.shape(imposter)

imp\_rows = dimensions[0]

FAR=np.zeros([11])

for i in range(imp\_rows):

score = imposter[i]

for j in range(11):

if score >= j/10: # threshold

FAR[j] +=1

for i in range(11):

FAR[i] = 100 \* FAR[i] / imp\_rows

# FRR

dimensions = np.shape(genuine)

gen\_rows = dimensions[0]

FRR=np.zeros(11)

for i in range(gen\_rows):

score = genuine[i]

for j in range(11):

if score < j/10: # threshold

FRR[j] +=1

for i in range(11):

FRR[i] = 100 \* FRR[i] / gen\_rows

# ROC curve

plt.plot(FAR, FRR)

plt.title("ROC Curve")

plt.xlabel("FAR (%)")

plt.ylabel("FRR (%)")

plt.xlim(0, 100)

plt.ylim(0, 100)

plt.rcParams["figure.autolayout"] = True

#(e) Threshold for EER and value of EER at threshold

threshold = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

plt.plot(threshold, FAR, 'r')

plt.plot(threshold, FRR, 'b')

plt.title("Finding Threshold from FAR and FRR curves")

plt.xlabel("Threshold")

plt.ylabel("Identification Rate (%)")

plt.legend(["FAR", "FRR"], loc=0, frameon=True)

#get intersection

line1 = LineString(np.column\_stack((threshold, FAR)))

line2 = LineString(np.column\_stack((threshold, FRR)))

intersection = line1.intersection(line2)

x, y = intersection.xy

plt.plot(\*intersection.xy, 'o')

plt.show()

print("Threshold value: ", x[0])

print("Equal Error Rate: ", y[0])

# ===============================================================================