Project 1

CSE 402 - Biometrics and Pattern Recognition
Instructor: Dr. Arun Ross
Due Date: October 7, 2022 (11:00 pm)
Total Points: 100

Note:

1. **You are permitted to discuss the following questions with others in the class. However, you *must* write up your *own* solutions to these questions. Any indication to the contrary will be considered an act of academic dishonesty. Copying from *any source* constitutes academic dishonesty.**
2. **A neatly typed report is expected (alternately, you can neatly handwrite the report and then scan it). The report, in PDF format, must be uploaded in D2L by October 7, 11:00 pm. Late submissions will not be graded. In your submission, please include the names of individuals you discussed this homework with and the list of external resources (e.g., websites, other books, articles, etc.) that you used to complete the assignment (if any).**
3. **When solving equations or reducing expressions you must explicitly show every step in your computation and/or include the code that was used to perform the computation. Missing steps or code will lead to a deduction of points.**
4. **Code developed as part of this assignment must be (a) included as an appendix to your report or inline with your solution (in the report), and (b) archived in a single zip file and uploaded in D2L. Including the code without the outputs or including the outputs without the code will result in deduction of points.**
5. **Please submit the report (PDF) and the code (Zip file) as two separate files in D2L.**

---

1. [10 points] Consider an experiment involving the face images from N different individuals. Assume that each individual, P_i, provides m_i face images, $i = 1, 2 \ldots N$. *Derive* expressions for the number of genuine scores and number of impostor scores that will be generated by a *symmetric* face matcher.

Ans) Total number of samples: $\Sigma\, m_i$

Genuine score generated by symmetric face matcher: $C(m_i, 2)*N$

Impostor score generated by symmetric face matcher: $C(N,2)* m_i^2$

Total scores: $C(N* m_i, 2)$

2. [15 points] Let B_1, B_2 and B_3 denote 3 different fingerprint matchers that are used to generate genuine and impostor match scores on a fixed set of fingerprint images. The mean (μ) and variance ($\square^2$) of the genuine and impostor score distributions resulting from the 3 different matchers are tabulated below.

Matcher	**Genuine**		**Impostor**	
	μ	$\square^2$	μ	$\square^2$
B_1	10	25	60	25
B_2	60	5	75	3
B_3	40	15	70	25

Based on the score statistics, determine which one of the three matchers has performed well and which one has performed the worst. Provide adequate numerical justification.

Ans) From the given data we can determine the best and worst matcher by using two equations :

I.

• d-prime: $\frac{\sqrt{2}|\mu_{gen} - \mu_{imp}|}{\sqrt{\sigma_{gen}^2 + \sigma_{imp}^2}}$

μ_{gen}: mean of genuine scores
σ_{gen}^2: variance of genuine scores

μ_{imp} :mean of impostor scores
σ_{imp}^2: variance of impostor scores

(Calculations done in python code (please check in Appendix))

The d-prime value of the B1 matcher = 10.0
The d-prime value of the B2 matcher = 7.5
The d-prime value of the B3 matcher = 6.71

The matcher that performed the best according to d-prime is: B1
The matcher that performed the worst according to d-prime is: B3

II.

• F-ratio: $\frac{|\mu_{gen} - \mu_{imp}|}{\sigma_{gen} + \sigma_{imp}}$

μ_{gen}: mean of genuine scores
σ_{gen}^2: variance of genuine scores
σ_{gen}: standard deviation

μ_{imp} :mean of impostor scores
σ_{imp}^2: variance of impostor scores
σ_{imp}: standard deviation

The f-ratio value of the B1 matcher = 5.0
The f-ratio value of the B2 matcher = 3.78
The f-ratio value of the B3 matcher = 3.38

The matcher that performed the best according to f-ratio is: B1
The matcher that performed the worst according to f-ratio is: B3

3. [15 points] Consider a biometric matcher that generates similarity scores in the range [0, 1]. Its genuine and impostor score distributions are as follows: $p(s\,\text{genuine}) = 3s^2$ and $p(s\,\text{impostor}) = 2 - 2s$. Suppose the following decision rule is employed: s is classified as a genuine score if $s \geq \square$; else it is classified as an impostor score. Here, $\square \in [0, 1]$.

- Plot the genuine and impostor distributions in a single graph.

Ans)

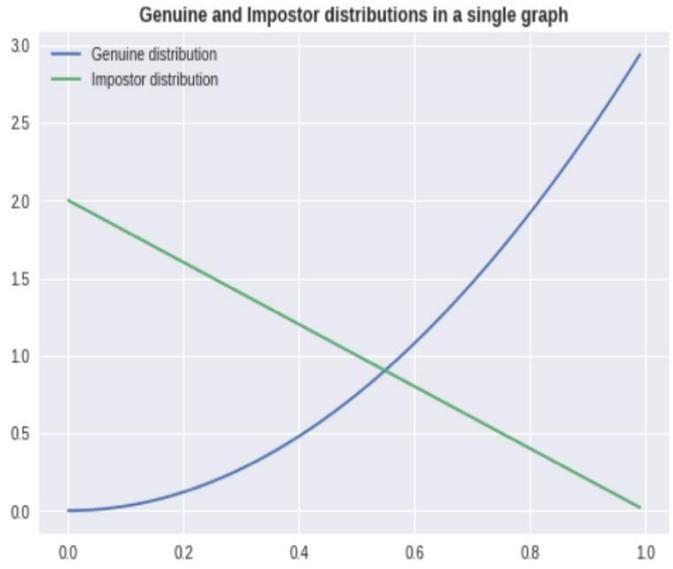


Figure 1

- If $\square = 0.2$, what is the FMR (i.e., FAR) and FNMR (i.e., FRR) of the biometric matcher?

Ans) **(Calculations done in python code (please check in Appendix))**

FMR of the biometric matcher with threshold value of 0.2 = 0.64
FNMR of the biometric matcher with threshold value of 0.2 = 0.008

- If $\square = 0.8$, what is the FMR (i.e., FAR) and FNMR (i.e., FRR) of the biometric matcher?

Ans) **(Calculations done in python code (please check in Appendix))**

FMR of the biometric matcher with threshold value of 0.8 = 0.04
FNMR of the biometric matcher with threshold value of 0.8 = 0.512

- Write a program to compute the DET curve based on these two distributions. Plot the DET curve.

Ans)

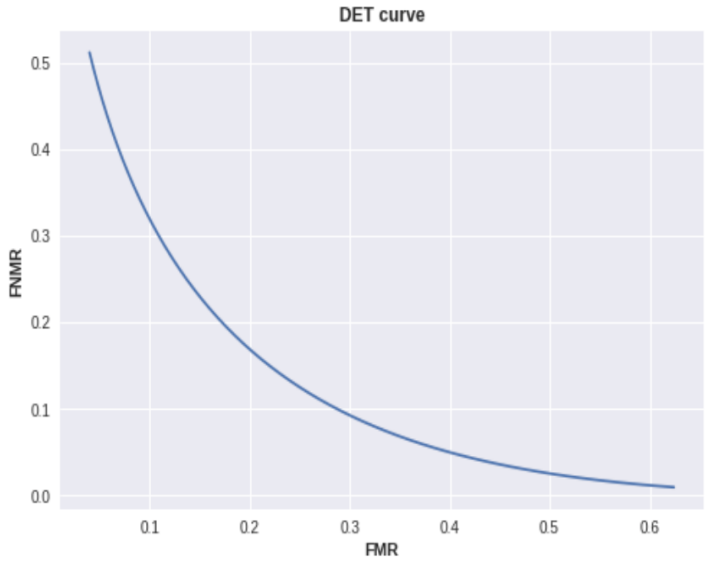


Figure 2

4. This exercise involves generating match score distributions and DET curves for two different modalities/matchers - fingerprint and hand. The fingerprint scores are *similarity-based*, while the hand scores are *distance-based*. The set of scores can be accessed here.

   (a) [5 points] Compute and report the mean and variance of the (a) genuine scores and (b) impostor scores for each matcher.

   Ans) **(Calculations done in python code (please check in Appendix))**

   Mean of finger genuine scores= 306.58
   Variance of finger genuine scores= 40825.04
   Mean of finger impostor scores= 7.97
   Variance of finger impostor scores= 90.81
   Mean of hand genuine scores= 50.64
   Variance of hand genuine scores= 1516.27
   Mean of hand impostor scores= 144.44
   Variance of hand impostor scores= 6925.66

   (b) [10 points] Compute and report the d-prime value for each matcher.

   Ans) **(Calculations done in python code (please check in Appendix))**

   d-prime value of finger matcher = 2.09
   d-prime value of hand matcher = 1.44

   (c) [10 points] For each matcher, plot the histogram of genuine and impostor scores in the same graph. So there will be two graphs - one for the fingerprint matcher and the other for the hand matcher.

   Ans) Fingerprint Matcher:

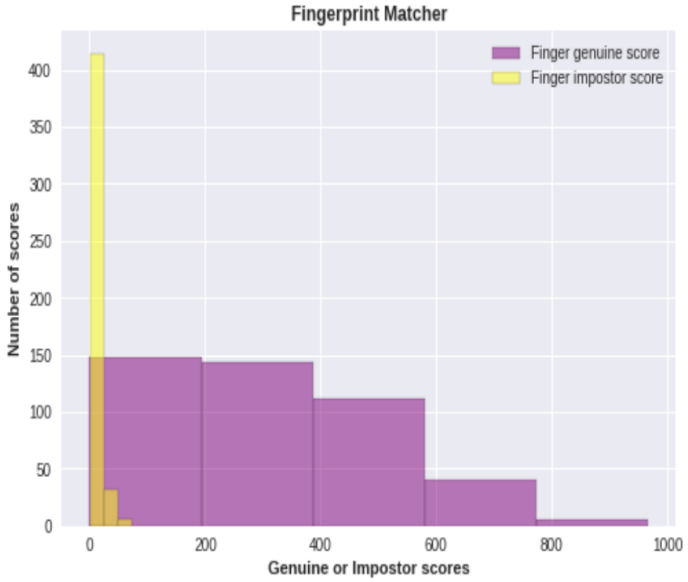


Figure 3

Hand Matcher:

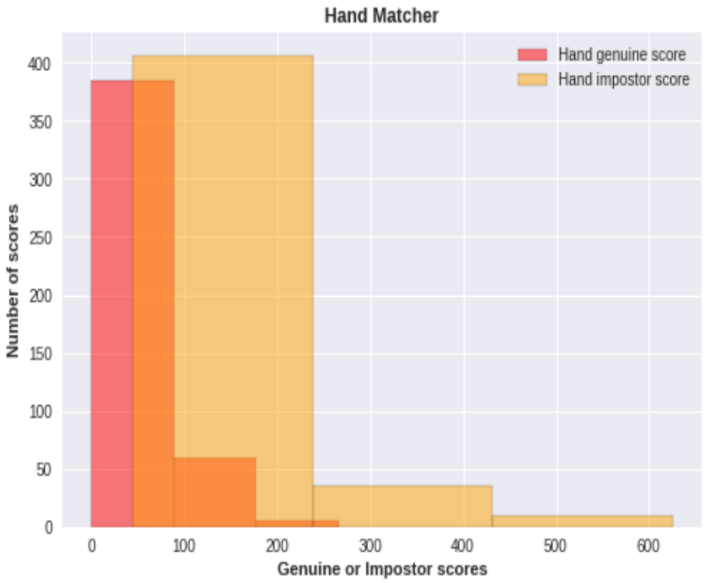


Figure 4

(d) [10 points] Write a program that inputs a threshold value, □, for each matcher and outputs the False Match Rate (FMR) and False Non-match Rate (FNMR) at that threshold. Use this program to compute the FMR and FNMR for the following scenarios:

i. Fingerprint Matcher: □ = 500

Ans) **(Calculations done in python code (please check in Appendix))**

False Match Rate Similarity = 0.0 %
False Non-Match Rate similarity = 80.22 %

ii. Hand Matcher: □ = 300

Ans) **(Calculations done in python code (please check in Appendix))**

False Match Rate Dissimilarity = 96.44 %
False Non-Match Rate Dissimilarity = 0.0 %

(e) [15 points] Based on the program designed in (4d), write another program that inputs a set of genuine scores and impostor scores and plots the Detection Error Tradeoff (DET) Curve. Use this program to plot the DET curve for both the matchers and report the Equal Error Rate (EER).

Ans) Fingerprint Matcher:

The equal error rate of finger matcher is = 0.08

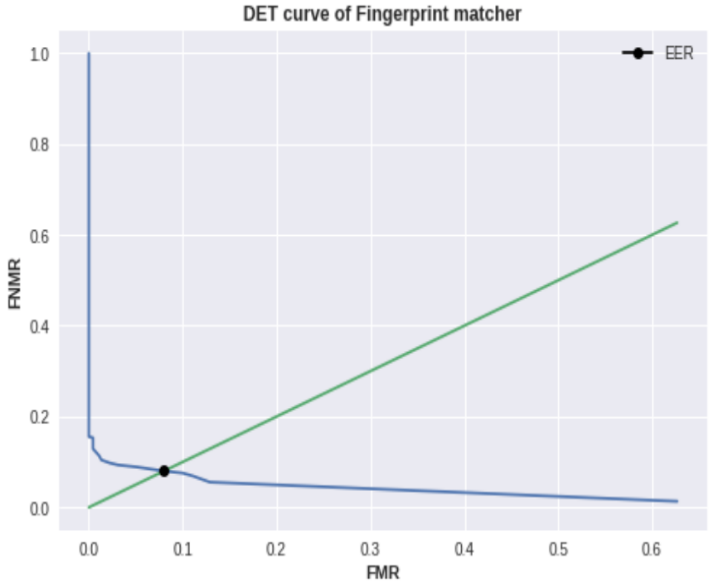


Figure 5

Hand Matcher:

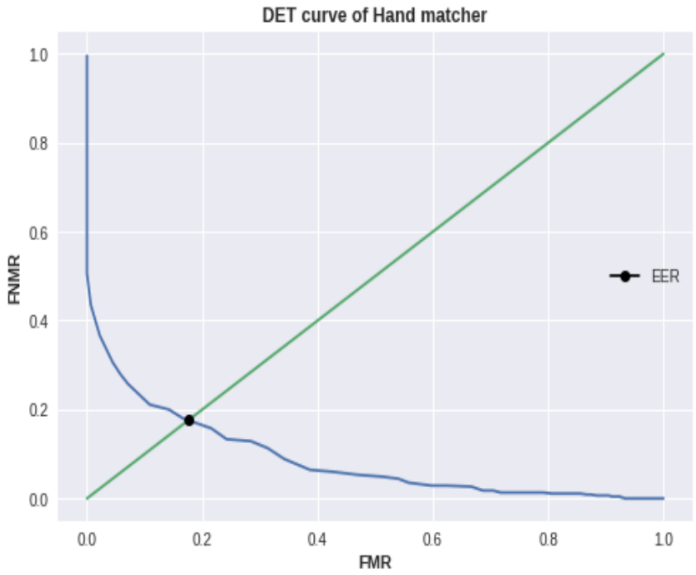


Figure 6

(f) [5 points] For each of the two matchers determine what the FNMR is at (a) FMR = 10%; (b) FMR = 5%; (c) FMR = 1%. You can determine these values from the DET curve.

Ans) Fingerprint Matcher:

FNMR value of fingerprint matcher at FMR = 10% is = 0.076
FNMR value of fingerprint matcher at FMR = 5% is = 0.089
FNMR value of fingerprint matcher at FMR = 1% is = 0.116

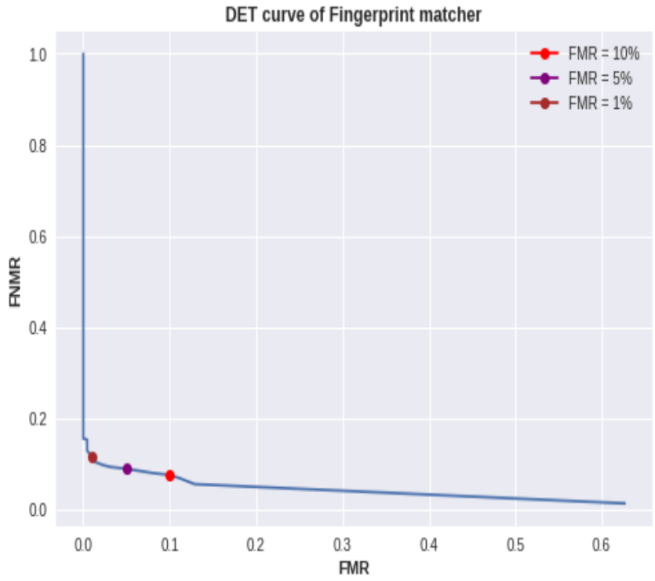


Figure 7

Hand Matcher:

FNMR value of hand matcher at FMR = 10% is = 0.222
FNMR value of hand matcher at FMR = 5% is = 0.296
FNMR value of hand matcher at FMR = 1% is = 0.421

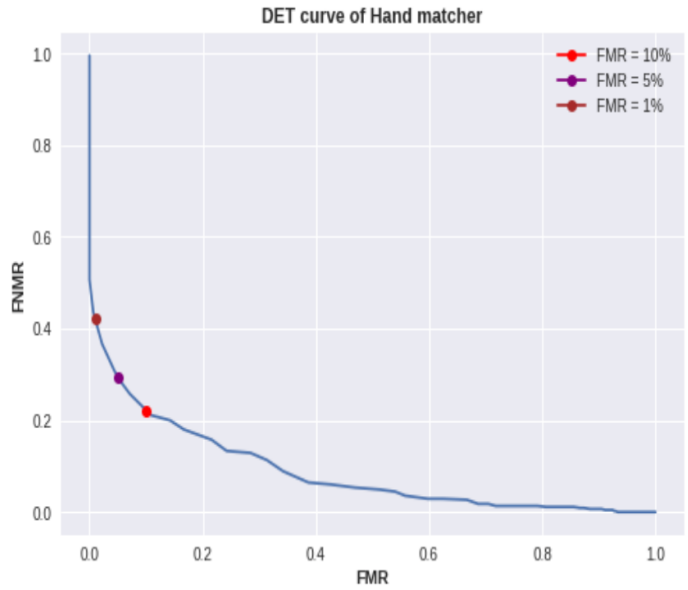


Figure 8

(g) [5 points] Which matcher, in your opinion, has performed well? **Justify your answer.**

Ans) According to my opinion fingerprint matcher has performed well than hand matcher because of its high d-prime value and less equal error rate (calculated in 4e and 4b).

Appendix

Appendix 1:

2) Let B1, B2 and B3 denote 3 different fingerprint matchers that are used to generate genuine and impostor match scores on a fixed set of fingerprint images. The mean ( ) and variance (2) of the genuine and impostor score distributions resulting from the 3 different matchers are tabulated below. Matcher Genuine Impostor

Matcher	mean(genuine)	variance(genuine)	mean(impostor)	variance(impostor)
B1	10	25	60	25
B2	60	5	75	3
B3	40	15	70	25

Based on the score statistics, determine which one of the three matchers has performed well and which one has performed the worst. Provide adequate numerical justification.

In [62]:
```
#2)

d_prime_dict = {"B1":[],"B2":[],"B3":[]}

d_prime_B1 = (math.sqrt(2)*(abs((10)-(60))))/(math.sqrt((25)+(25)))
print ("The d-prime value of the B1 matcher =",round(d_prime_B1,2))
d_prime_dict["B1"].append (d_prime_B1)

d_prime_B2 = (math.sqrt(2)*(abs((60)-(75))))/(math.sqrt((5)+(3)))
print ("The d-prime value of the B2 matcher =",round(d_prime_B2,2))
d_prime_dict["B2"].append(d_prime_B2)

d_prime_B3 = (math.sqrt(2)*(abs((40)-(70))))/(math.sqrt((15)+(25)))
print ("The d-prime value of the B3 matcher =",round(d_prime_B3,2))
d_prime_dict["B3"].append(d_prime_B3)
```

```
The d-prime value of the B1 matcher = 10.0
The d-prime value of the B2 matcher = 7.5
The d-prime value of the B3 matcher = 6.71
```

In [63]:
```
best_matcher = max(d_prime_dict, key = d_prime_dict.get)
print("The matcher that performed the best according to d-prime is:", best_matcher)

worst_matcher = min (d_prime_dict, key = d_prime_dict.get)
print("The matcher that performed the worst according to d-prime is:", worst_matcher)
```

```
The matcher that performed the best according to d-prime is: B1
The matcher that performed the worst according to d-prime is: B3
```

In [64]:
```
f_ratio_dict = {"B1":[],"B2":[],"B3":[]}

f_ratio_B1 = (abs((10)-(60)))/((math.sqrt(25))+(math.sqrt(25)))
print ("The f-ratio value of the B1 matcher =",round(f_ratio_B1,2))
f_ratio_dict["B1"].append (f_ratio_B1)

f_ratio_B2 = (abs((60)-(75)))/((math.sqrt(5))+(math.sqrt(3)))
print ("The f-ratio value of the B2 matcher =",round(f_ratio_B2,2))
f_ratio_dict["B2"].append (f_ratio_B2)

f_ratio_B3 = (abs((40)-(70)))/((math.sqrt(15))+(math.sqrt(25)))
print ("The f-ratio value of the B3 matcher =",round(f_ratio_B3,2))
f_ratio_dict["B3"].append (f_ratio_B3)
```

```
The f-ratio value of the B1 matcher = 5.0
The f-ratio value of the B2 matcher = 3.78
The f-ratio value of the B3 matcher = 3.38
```

In [65]:
```
best_matcher = max(f_ratio_dict, key = f_ratio_dict.get)
print("The matcher that performed the best according to f-ratio is:", best_matcher)

worst_matcher = min(f_ratio_dict, key = f_ratio_dict.get)
print("The matcher that performed the worst according to f-ratio is:", worst_matcher)
```

```
The matcher that performed the best according to f-ratio is: B1
The matcher that performed the worst according to f-ratio is: B3
```

Appendix 2:

3) Consider a biometric matcher that generates similarity scores in the range [0, 1]. Its genuine and impostor score distributions are as follows: p(s|genuine)=3s^2 and p(s|impostor)=2-2s. Suppose the following decision rule is employed: s is classified as a genuine score if s>=n; else it is classified as an impostor score. Here,n E [0, 1].

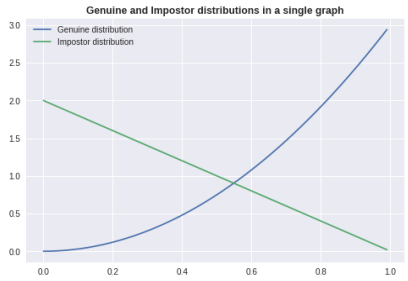
a) Plot the genuine and impostor distributions in a single graph.

b) If n = 0.2, what is the FMR (i.e., FAR) and FNMR (i.e., FRR) of the biometric matcher?

c) If n = 0.8, what is the FMR (i.e., FAR) and FNMR (i.e., FRR) of the biometric matcher?

d) Write a program to compute the DET curve based on these two distributions. Plot the DET curve.

In [66]:
```
# 3a)

genuine_distribution=[]
impostor_distribution =[]
temp =[]
s = 0
while s < 1:
    impostor = 2-2*s
    genuine = 3*s**2

    genuine_distribution.append(genuine)
    impostor_distribution.append(impostor)
    temp.append(s)
    s+=0.01

plt.style.use('seaborn')
plt.title("Genuine and Impostor distributions in a single graph",fontweight='bold')
plt.plot(temp,genuine_distribution, label='Genuine distribution')
plt.plot(temp,impostor_distribution, label='Impostor distribution')
plt.legend()
plt.show()
```



In [67]:
```
#3b)

def genuine_distribution(s):

    return 3*s**2

def impostor_distribution(s):

    return 2-2*s


threshold_value = 0.2

fmr_1,error_fmr = integrate.quad(impostor_distribution,threshold_value,1)

fnmr_1,error_fnmr = integrate.quad(genuine_distribution,0,threshold_value)

print("FMR of the biometric matcher with threshold value of 0.2 = ",round(fmr_1,3))
print("FNMR of the biometric matcher with threshold value of 0.2 = ",round(fnmr_1,3))
```

```
FMR of the biometric matcher with threshold value of 0.2 =  0.64
FNMR of the biometric matcher with threshold value of 0.2 =  0.008
```

In [68]:
```
#3c)

def genuine_distribution(s):

    return 3*s**2

def impostor_distribution(s):

    return 2-2*s


threshold_value = 0.8

fmr_2,error_fmr = integrate.quad(impostor_distribution,threshold_value,1)

fnmr_2,error_fnmr = integrate.quad(genuine_distribution,0,threshold_value)

print("FMR of the biometric matcher with threshold value of 0.8 = ",round(fmr_2,3))
print("FNMR of the biometric matcher with threshold value of 0.8 = ",round(fnmr_2,3))
```

```
FMR of the biometric matcher with threshold value of 0.8 =  0.04
FNMR of the biometric matcher with threshold value of 0.8 =  0.512
```

```
#3d)

initial_value = 0.2
final_value = 0.8
fmr_values=[]
fnmr_values=[]

while initial_value<=final_value:

    initial_value +=0.01

    fmr,error_fmr = integrate.quad(impostor_distribution,initial_value,1)

    fnmr,error_fnmr = integrate.quad(genuine_distribution,0,initial_value)


    fmr_values.append(fmr)

    fnmr_values.append(fnmr)
```

```
plt.plot(fmr_values,fnmr_values)
plt.title("DET curve")
plt.show()
```

Appendix 3:

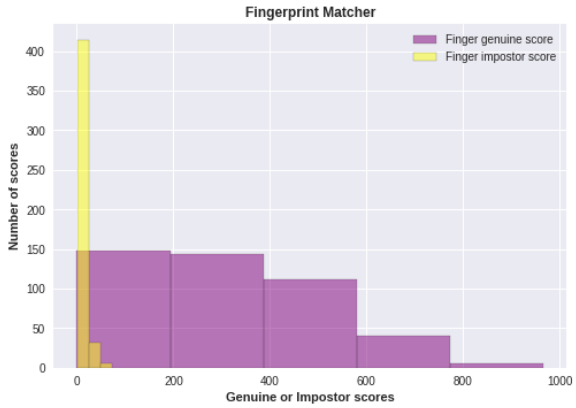
In [71]:
```
# 4a

finger_genuine_file = open("finger_genuine.score")
finger_impostor_file = open("finger_impostor.score")
hand_genuine_file = open("hand_genuine.score")
hand_impostor_file = open("hand_impostor.score")

finger_genuine_file_list = finger_genuine_file.readlines()
finger_impostor_file_list = finger_impostor_file.readlines()
hand_genuine_file_list = hand_genuine_file.readlines()
hand_impostor_file_list = hand_impostor_file.readlines()

for values in range(len(finger_genuine_file_list)):
    finger_genuine_file_list[values] = (float(finger_genuine_file_list[values]))

for values in range(len(finger_impostor_file_list)):
    finger_impostor_file_list[values] = (float(finger_impostor_file_list[values]))

for values in range(len(hand_genuine_file_list)):
    hand_genuine_file_list[values] = (float(hand_genuine_file_list[values]))


for values in range(len(hand_impostor_file_list)):
    hand_impostor_file_list[values] = (float(hand_impostor_file_list[values]))

mean_finger_genuine_file_list = np.mean(finger_genuine_file_list)
variance_finger_genuine_file_list= np.var(finger_genuine_file_list)

mean_finger_impostor_file_list = np.mean(finger_impostor_file_list)
variance_finger_impostor_file_list = np.var(finger_impostor_file_list)

mean_hand_genuine_file_list = np.mean(hand_genuine_file_list)
variance_hand_genuine_file_list = np.var(hand_genuine_file_list)

mean_hand_impostor_file_list = np.mean(hand_impostor_file_list)
variance_hand_impostor_file_list = np.var(hand_impostor_file_list)


print("Mean of finger genuine scores=",round(mean_finger_genuine_file_list,2))
print("Variance of finger genuine scores=",round(variance_finger_genuine_file_list,2))

print("Mean of finger impostor scores=",round(mean_finger_impostor_file_list,2))
print("Varinace of finger impostor scores=",round(variance_finger_impostor_file_list,2)

print("Mean of hand genuine scores=",round(mean_hand_genuine_file_list,2))
print("Variance of hand genuine scores=",round(variance_hand_genuine_file_list,2))

print("Mean of hand impostor scores=",round(mean_hand_impostor_file_list,2))
print("Variance of hand impostor scores=",round(variance_hand_impostor_file_list,2))
```

```
Mean of finger genuine scores= 306.58
Variance of finger genuine scores= 40825.04
Mean of finger impostor scores= 7.97
Varinace of finger impostor scores= 90.81
Mean of hand genuine scores= 50.64
Variance of hand genuine scores= 1516.27
Mean of hand impostor scores= 144.44
Variance of hand impostor scores= 6925.66
```

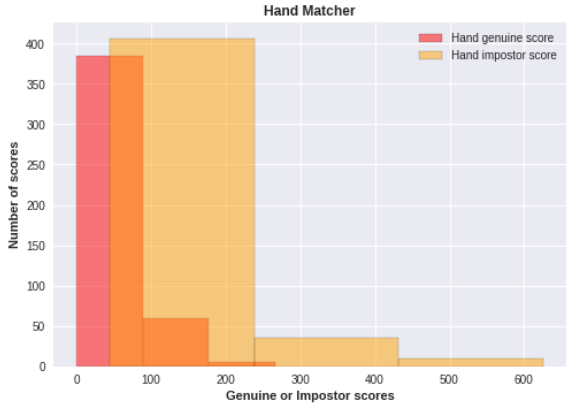
In [72]:
```
#4b

d_prime_finger_matcher = math.sqrt(2)*(abs(mean_finger_genuine_file_list)
-(mean_finger_impostor_file_list))/(math.sqrt((variance_finger_genuine_file_list)
                                              +(variance_finger_impostor_file_list)))
print ("d-prime value of finger matcher =",round(d_prime_finger_matcher,2))

d_prime_hand_matcher = math.sqrt(2)*(abs((mean_hand_genuine_file_list)
-(mean_hand_impostor_file_list)))/(math.sqrt((variance_hand_genuine_file_list)
                                            +(variance_hand_impostor_file_list)))
print ("d-prime value of hand matcher =",round(d_prime_hand_matcher,2))
```

```
d-prime value of finger matcher = 2.09
d-prime value of hand matcher = 1.44
```

In [73]:
```
#4c

plt.hist(finger_genuine_file_list,label = "Finger genuine score", color='purple',
         edgecolor = 'black',alpha =0.5, bins=5)

plt.hist(finger_impostor_file_list, label = "Finger impostor score", color='yellow',
         edgecolor = 'black',alpha =0.5, bins=3)

plt.xlabel('Genuine or Impostor scores', fontweight='bold')
plt.ylabel('Number of scores',fontweight='bold')
plt.title('Fingerprint Matcher',fontweight='bold')
plt.legend()

plt.show()
```



In [74]:
```
plt.hist(hand_genuine_file_list, label = "Hand genuine score", color='red',
         edgecolor = 'black',alpha =0.5,bins =3)

plt.hist(hand_impostor_file_list, label = "Hand impostor score", color='orange',
         edgecolor = 'black',alpha =0.5, bins =3)

plt.xlabel('Genuine or Impostor scores',fontweight='bold')
plt.ylabel('Number of scores',fontweight='bold')
plt.title('Hand Matcher',fontweight='bold')
plt.legend()


plt.show()
```



In [75]:
```
#4d

threshold = 500
false_match_rate_finger = 0
false_non_match_rate_finger = 0

for i in range(len(finger_genuine_file_list)):
    if finger_genuine_file_list[i] < threshold:
        false_non_match_rate_finger+=1


for count in range(len(finger_impostor_file_list)):
    if finger_impostor_file_list[count] >= threshold:
        false_match_rate_finger+=1



fmr_similarity=false_match_rate_finger/len(finger_impostor_file_list)
fnmr_similarity=false_non_match_rate_finger/len(finger_genuine_file_list)

print("False Match Rate Similarity = ",round(fmr_similarity*100,2),"%")
print("False Non-Match Rate similarity = ",round(fnmr_similarity*100,2),"%")
```

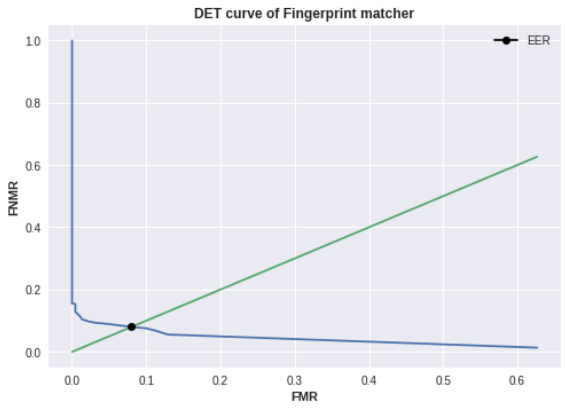
```
False Match Rate Similarity =  0.0 %
False Non-Match Rate similarity =  80.22 %
```

In [76]:
```
threshold = 300

false_match_rate_hand = 0
false_non_match_rate_hand = 0


for j in range (len(hand_genuine_file_list)):
    if hand_genuine_file_list[j] > threshold:
        false_non_match_rate_hand+=1


for count in range (len(hand_impostor_file_list)):
    if hand_impostor_file_list[count] <= threshold:
        false_match_rate_hand+=1


fmr_disimilarity=false_match_rate_hand/len(hand_impostor_file_list)
fnmr_disimilarity=false_non_match_rate_hand/len(hand_genuine_file_list)


print("False Match Rate Dissimilarity = ",round(fmr_disimilarity*100,2),"%")
print("False Non-Match Rate Dissimilarity = ",round(fnmr_disimilarity*100,2),"%")
```

```
False Match Rate Dissimilarity =  96.44 %
False Non-Match Rate Dissimilarity =  0.0 %
```

In [77]:
```
#4e

finger_maximum_value = int(max(max(finger_genuine_file_list),
                               max(finger_impostor_file_list)))
finger_minimum_value = int(min(min(finger_genuine_file_list),
                               min(finger_impostor_file_list)))

initial_value = finger_minimum_value

fmr_finger_similariy = []
fnmr_finger_similarity = []

for i in range (finger_minimum_value , finger_maximum_value,5):

    initial_value += 5

    false_match_rate_finger = 0
    false_non_match_rate_finger = 0

    for j in range(len(finger_genuine_file_list)):
        if finger_genuine_file_list[j] < initial_value:
            false_non_match_rate_finger+=1

    for count in range(len(finger_impostor_file_list)):
        if finger_impostor_file_list[count] >= initial_value:
            false_match_rate_finger+=1

    fmr_similarity=false_match_rate_finger/len(finger_impostor_file_list)
    fnmr_similarity=false_non_match_rate_finger/len(finger_genuine_file_list)

    fmr_finger_similariy.append(fmr_similarity)

    fnmr_finger_similarity.append(fnmr_similarity)
```

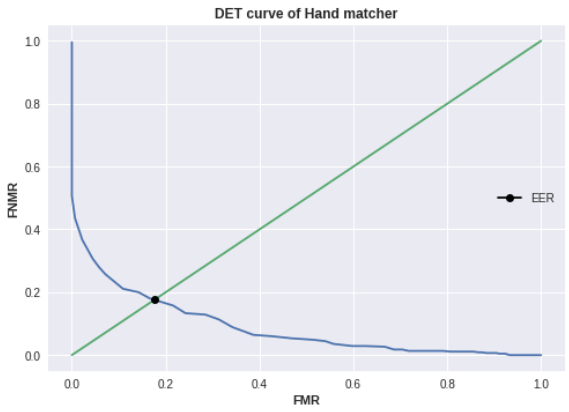
In [78]:
```
plt.plot(fmr_finger_similariy,fnmr_finger_similarity)
plt.plot(fmr_finger_similariy, fmr_finger_similariy)


first_line_finger = LineString(np.column_stack((fmr_finger_similariy,
                                                fnmr_finger_similarity)))
second_line_finger = LineString(np.column_stack((fmr_finger_similariy,
                                                 fmr_finger_similariy)))
intersection_finger = first_line_finger.intersection(second_line_finger)
x_finger, y_finger = intersection_finger.xy
plt.plot(x_finger,y_finger,'r-o',color ='black', label = "EER")

plt.legend()
plt.xlabel('FMR',fontweight='bold')
plt.ylabel('FNMR',fontweight='bold')
plt.title("DET curve of Fingerprint matcher",fontweight='bold')
plt.show()
print("The equal error rate of finger matcher is = ",y_finger[0])
```




```
The equal error rate of finger matcher is =  0.08
```

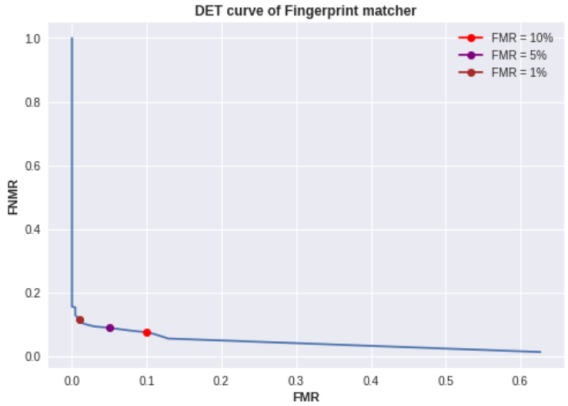
In [79]:
```
#4e)

hand_maximum_value = int(max(max(hand_genuine_file_list),
                             max(hand_impostor_file_list)))
hand_minimum_value = int(min(min(hand_genuine_file_list),
                             min(hand_impostor_file_list)))

initial_value = hand_minimum_value

fmr_hand_disimilariy = []
fnmr_hand_disimilarity = []

for i in range (hand_minimum_value , hand_maximum_value,5):

    initial_value += 5

    false_match_rate_hand = 0
    false_non_match_rate_hand = 0

    for j in range (len(hand_genuine_file_list)):
        if hand_genuine_file_list[j] > initial_value:
            false_non_match_rate_hand+=1


    for count in range (len(hand_impostor_file_list)):
        if hand_impostor_file_list[count] <= initial_value:
            false_match_rate_hand+=1

    fmr_disimilarity=false_match_rate_hand/len(hand_impostor_file_list)
    fnmr_disimilarity=false_non_match_rate_hand/len(hand_genuine_file_list)

    fmr_hand_disimilariy.append(fmr_disimilarity)

    fnmr_hand_disimilarity.append(fnmr_disimilarity)
```

In [80]:
```
plt.plot(fmr_hand_disimilariy,fnmr_hand_disimilarity)
plt.plot(fmr_hand_disimilariy, fmr_hand_disimilariy)


first_line_hand = LineString(np.column_stack((fmr_hand_disimilariy,
                                              fnmr_hand_disimilarity)))
second_line_hand = LineString(np.column_stack((fmr_hand_disimilariy,
                                               fmr_hand_disimilariy)))
intersection_hand = first_line_hand.intersection(second_line_hand)
x_hand, y_hand = intersection_hand.xy
plt.plot(x_hand,y_hand,'r-o',color ='black', label = "EER")

plt.legend()
plt.xlabel('FMR',fontweight='bold')
plt.ylabel('FNMR',fontweight='bold')
plt.title("DET curve of Hand matcher",fontweight='bold')
plt.show()
print("The equal error rate of hand matcher is = ",round(y_hand[0],3))
```



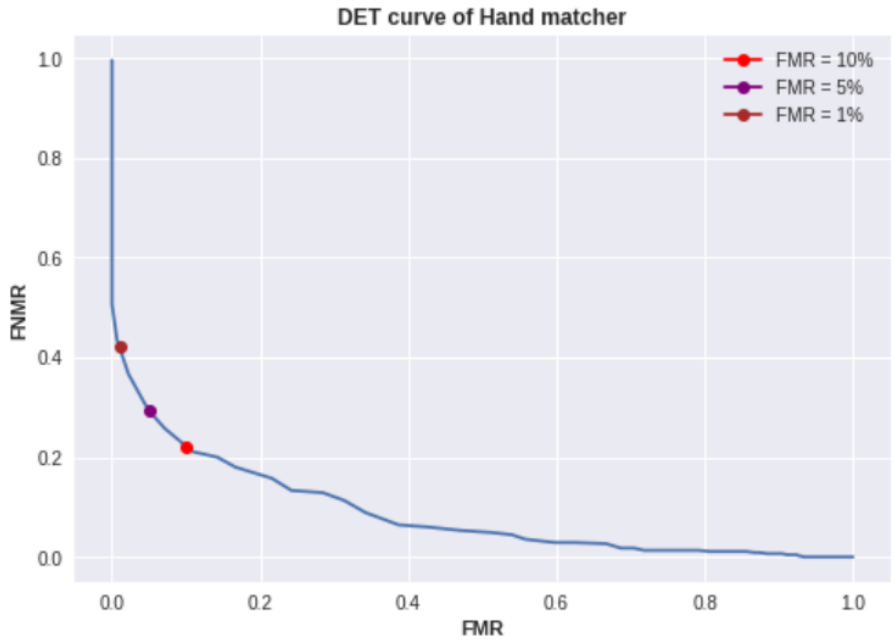

```
The equal error rate of hand matcher is =  0.176
```

In [81]:
```
#4f)
plt.plot(fmr_finger_similariy,fnmr_finger_similarity)


first_line_ten_percent = LineString(np.column_stack((fmr_finger_similariy,fnmr_finger_similarity)))
second_line_ten_percent = LineString(np.column_stack(([0.1,0.1],[0,1])))
intersection_ten_percent = first_line_ten_percent.intersection(second_line_ten_percent)
x_ten_percent, y_ten_percent = intersection_ten_percent.xy
plt.plot(x_ten_percent, y_ten_percent,'r-o',color ='red', label ='FMR = 10%')

first_line_five_percent = LineString(np.column_stack((fmr_finger_similariy,fnmr_finger_similarity)))
second_line_five_percent = LineString(np.column_stack(([0.05,0.05],[0,1])))
intersection_five_percent = first_line_five_percent.intersection(second_line_five_percent)
x_five_percent, y_five_percent = intersection_five_percent.xy
plt.plot(x_five_percent, y_five_percent,'r-o',color ='purple', label ='FMR = 5%')

first_line_one_percent = LineString(np.column_stack((fmr_finger_similariy,fnmr_finger_similarity)))
second_line_one_percent = LineString(np.column_stack(([0.01,0.01],[0,1])))
intersection_one_percent = first_line_one_percent.intersection(second_line_one_percent)
x_one_percent, y_one_percent = intersection_one_percent.xy
plt.plot(x_one_percent, y_one_percent,'r-o',color ='brown', label ='FMR = 1%')

plt.legend()
plt.xlabel('FMR',fontweight='bold')
plt.ylabel('FNMR',fontweight='bold')
plt.title("DET curve of Fingerprint matcher",fontweight='bold')
plt.show()
print("FNMR value of fingerprint matcher at FMR = 10% is =", round(y_ten_percent[0],3))
print("FNMR value of fingerprint matcher at FMR = 5% is =", round(y_five_percent[0],3))
print("FNMR value of fingerprint matcher at FMR = 1% is =", round(y_one_percent[0],3))
```




```
FNMR value of fingerprint matcher at FMR = 10% is = 0.076
FNMR value of fingerprint matcher at FMR = 5% is = 0.089
FNMR value of fingerprint matcher at FMR = 1% is = 0.116
```

In [82]:
```
#4f)

plt.plot(fmr_hand_disimilariy,fnmr_hand_disimilarity)


first_line_ten_percent = LineString(np.column_stack((fmr_hand_disimilariy,fnmr_hand_disimilarity)))
second_line_ten_percent = LineString(np.column_stack(([0.1,0.1],[0,1])))
intersection_ten_percent = first_line_ten_percent.intersection(second_line_ten_percent)
x_ten_percent, y_ten_percent = intersection_ten_percent.xy
plt.plot(x_ten_percent, y_ten_percent,'r-o',color ='red', label ='FMR = 10%')


first_line_five_percent = LineString(np.column_stack((fmr_hand_disimilariy,fnmr_hand_disimilarity)))
second_line_five_percent = LineString(np.column_stack(([0.05,0.05],[0,1])))
intersection_five_percent = first_line_five_percent.intersection(second_line_five_percent)
x_five_percent, y_five_percent = intersection_five_percent.xy
plt.plot(x_five_percent, y_five_percent,'r-o',color ='purple', label ='FMR = 5%')


first_line_one_percent = LineString(np.column_stack((fmr_hand_disimilariy,fnmr_hand_disimilarity)))
second_line_one_percent = LineString(np.column_stack(([0.01,0.01],[0,1])))
intersection_one_percent = first_line_one_percent.intersection(second_line_one_percent)
x_one_percent, y_one_percent = intersection_one_percent.xy
plt.plot(x_one_percent, y_one_percent,'r-o',color ='brown', label ='FMR = 1%')


plt.legend()
plt.xlabel('FMR',fontweight='bold')
plt.ylabel('FNMR',fontweight='bold')
plt.title("DET curve of Hand matcher",fontweight='bold')
plt.show()

print("FNMR value of hand matcher at FMR = 10% is =", round(y_ten_percent[0],3))
print("FNMR value of hand matcher at FMR = 5% is =", round(y_five_percent[0],3))
print("FNMR value of hand matcher at FMR = 1% is =", round(y_one_percent[0],3))
```




```
FNMR value of hand matcher at FMR = 10% is = 0.222
FNMR value of hand matcher at FMR = 5% is = 0.296
FNMR value of hand matcher at FMR = 1% is = 0.421
```