Amirkabir University of Technology
(Tehran Polytechnic)
Department of Computer Engineering

Statistical Pattern Recognition

Assignment II

Soroush Mahdi
99131050

supervised by
Dr. Mohammad Rahmati

December 30, 2020

Contents

1 Do You Think You May Have COVID-19?

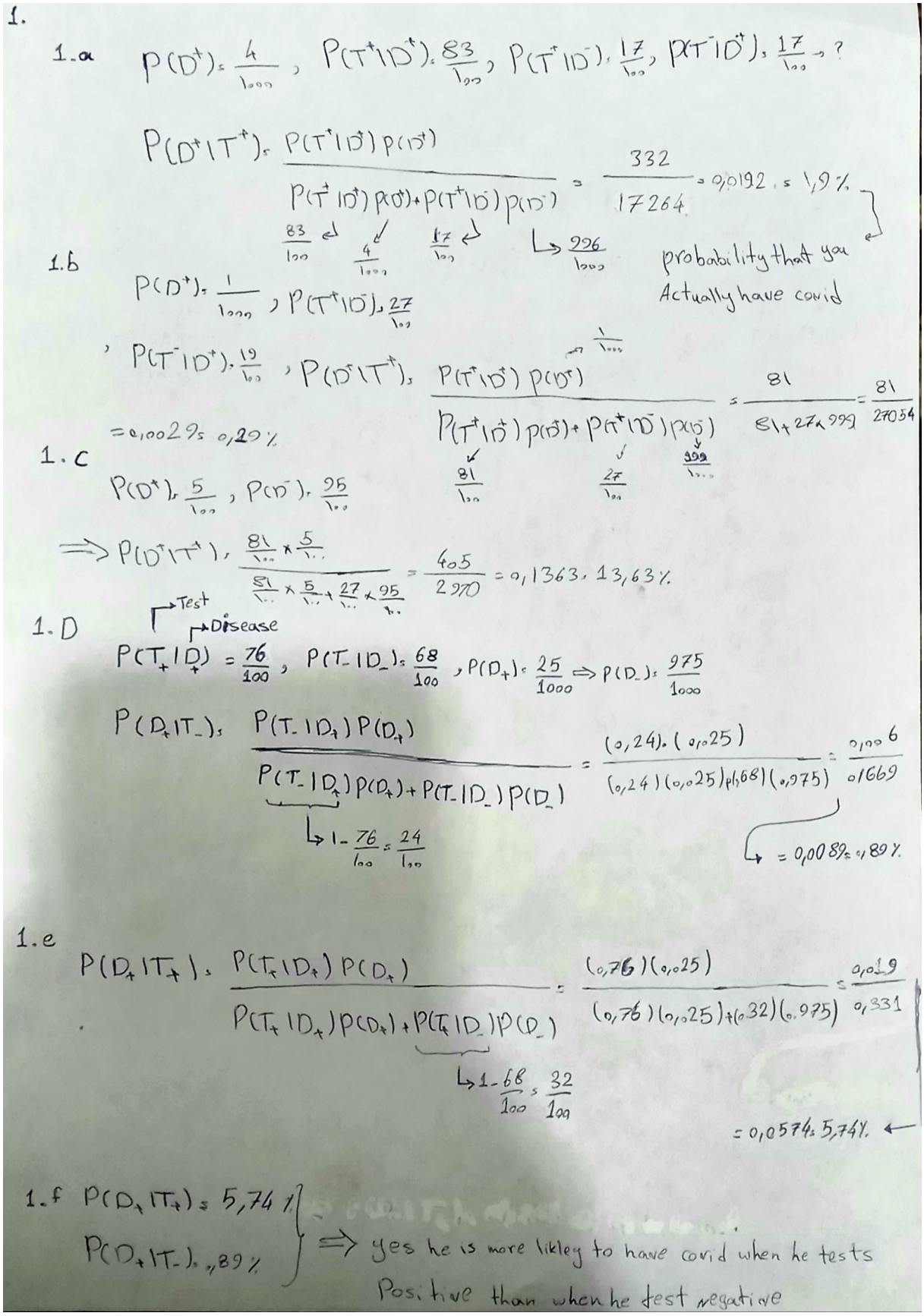


Figure 1: a-f

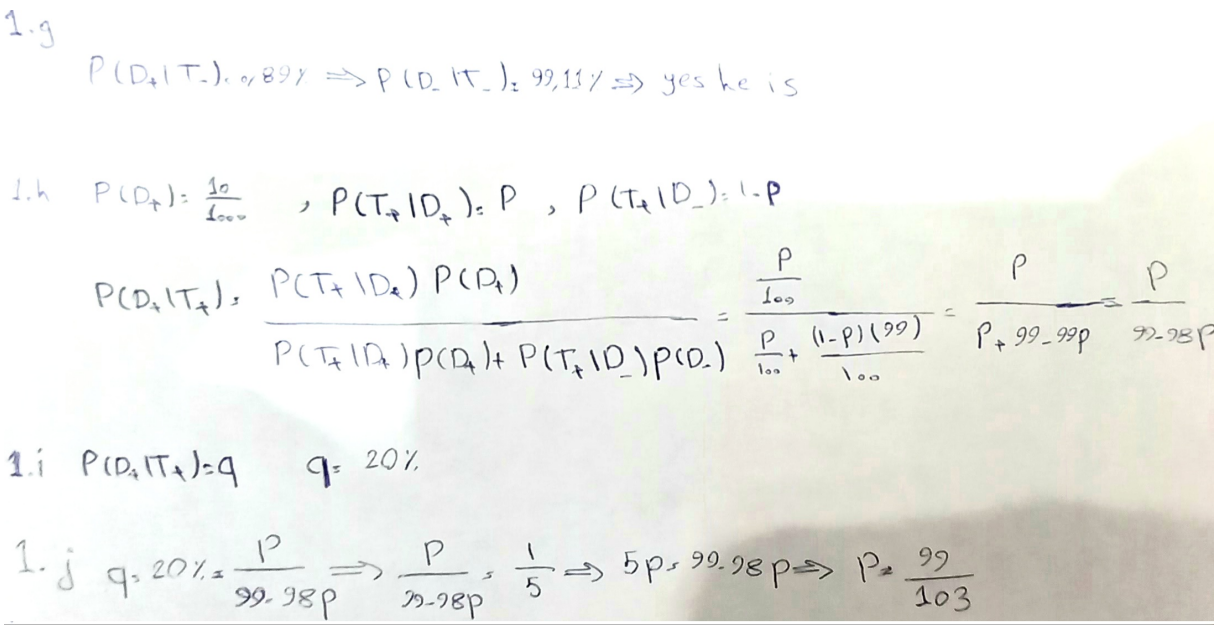


Figure 2: g-j

2 Making Decisions using Bayes Decision Rule

2.1 a

```
#mu1 = [1, 2]
mu2 = [-1, -3]
cov1 = [[1.8, -0.7], [-0.7, 1.8]]
cov2 = [[1.5, 0.3], [0.3, 1.5]]

x1,y1 = np.random.multivariate_normal(mu1, cov1,1000).T
x2,y2 = np.random.multivariate_normal(mu2, cov2, 1000).T

xx1, yy1 = np.meshgrid(x1,y1)
pos1 = np.dstack((xx1,yy1))
rv1 = multivariate_normal(mean=mu1, cov=cov1)
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.contourf(xx1, yy1, rv1.pdf(pos1))

fig2.savefig('2a1.png')

xx2, yy2 = np.meshgrid(x2,y2)
pos2 = np.dstack((xx2,yy2))
rv2 = multivariate_normal(mean=mu2, cov=cov2)
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.contourf(xx2, yy2, rv2.pdf(pos2))

fig2.savefig('2a2.png')
```

Listing 1: 2a

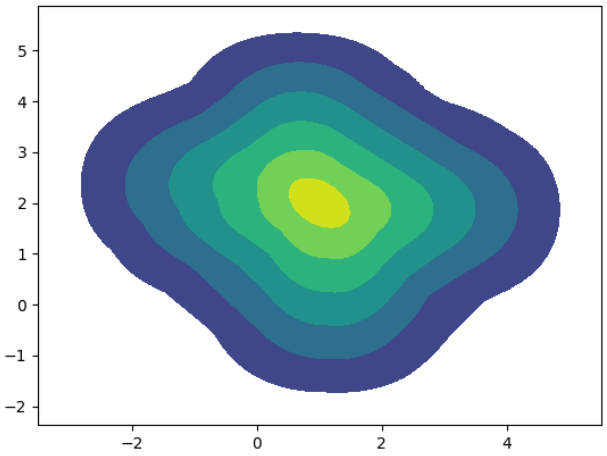


Figure 3: class 1

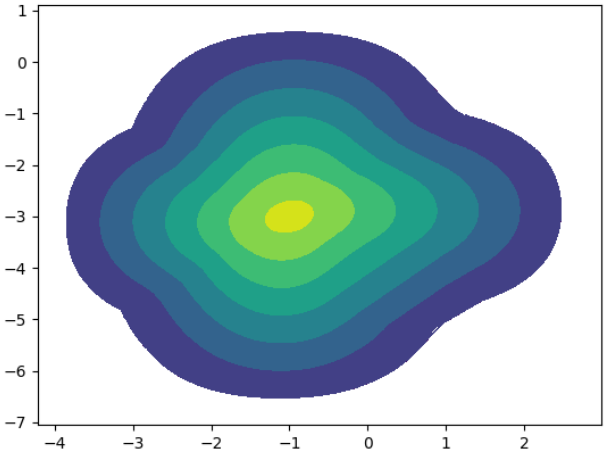


Figure 4: class 2

2.2 b

```
x = np.linspace(-30, 30, 15000)
plt.plot(x,-0.4*x-0.5 )

Y, X = np.mgrid[-30:30:300j, -30:30:300j]
plt.contour(X, Y, (63*X**2 - 128*X +36*Y**2 + 28*X*Y -172*Y + 236)/55 -
            (25*X**2 + 20 * X + 25 *Y**2 -10*X*Y+140*Y+220)/36 +0.24   , levels=[0])
plt.grid()

plt.savefig('2b.png')
```

Listing 2: 2b

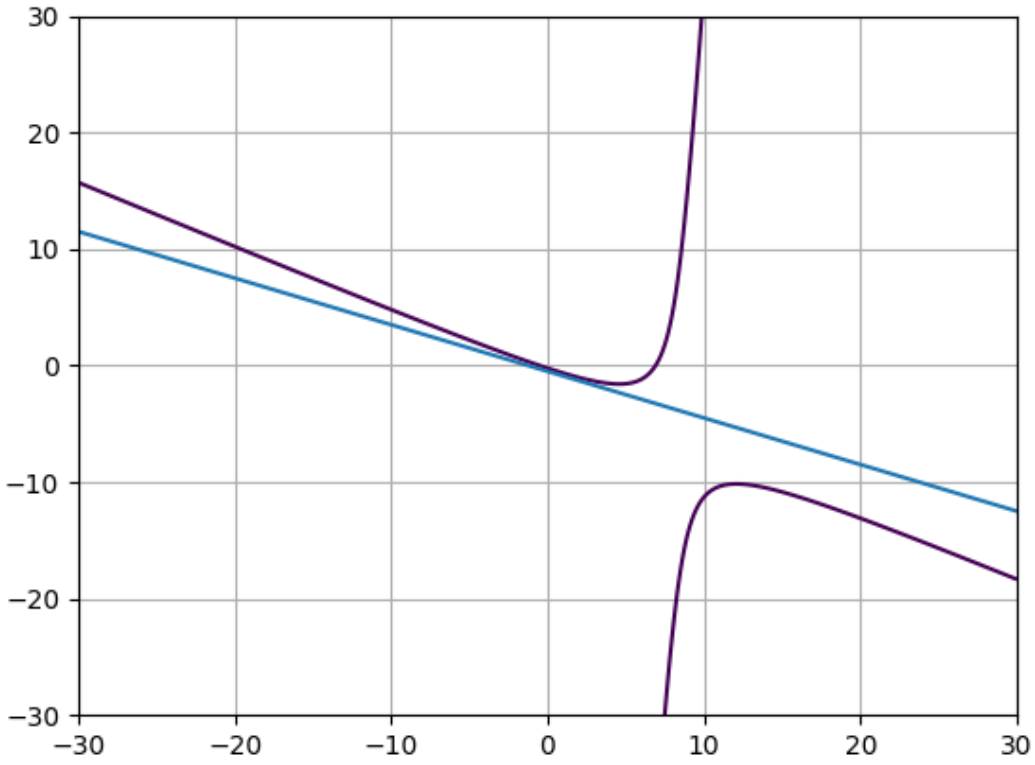


Figure 5: decision boundary

2.3 c

```
def MDC (x):
    return 2*x[0] + 5 * x[1] + 0.4

def bayes(x):
    mu1 = np.array([1, 2])
    mu2 = np.array([-1, -3])
    cov1 = np.array([[1.8, -0.7], [-0.7, 1.8]] )
    cov2 = np.array([[1.5, 0.3], [0.3, 1.5]] )

    return -0.5 * ((x-mu1).T)@np.linalg.inv(cov1)@(x-mu1)-0.5*np.log(np.abs(np.linalg
    .det(cov1))) - (-0.5 * ((x-mu2).T)@np.linalg.inv(cov2)@(x-mu2)-0.5*np.log(np.abs(
    np.linalg.det(cov2))))


mu1 = [1, 2]
mu2 = [-1, -3]
cov1 = [[1.8, -0.7], [-0.7, 1.8]]
cov2 = [[1.5, 0.3], [0.3, 1.5]]

x0 = np.random.multivariate_normal(mu1, cov1,1000)
x1 = np.random.multivariate_normal(mu2, cov2, 1000)

mdc_err = 0
bayes_err = 0

for i in x0:
    if MDC(i)<0:
        mdc_err+=1

```

```
    if bayes(i)<0:
        bayes_err +=1

for i in x1:
    if MDC(i)>=0:
        mdc_err+=1

    if bayes(i)>=0:
        bayes_err +=1

print('MDC error: ',mdc_err/2000)
print('bayes error: ',bayes_err/2000)

#MDC error:  0.018
#bayes error:  0.012
```

Listing 3: 2c

2.4 d

```
TP_bayes_cost = 0
TP_bayes = 0

for i in x0:
    if bayes(i)>=np.log(3):
        TP_bayes_cost +=1

    if bayes(i)>=0:
        TP_bayes +=1


recall_cost = TP_bayes_cost/2000
recall_bayes = TP_bayes/2000

FP_cost = 0
FP_bayes = 0

for i in x1:
    if bayes(i)>=np.log(3):
        FP_cost+=1

    if bayes(i)>=0:
        FP_bayes +=1

precision_cost = TP_bayes_cost / (TP_bayes_cost + FP_cost)
precision_bayes = TP_bayes/(TP_bayes + FP_bayes)

fscore_cost = 2*precision_cost*recall_cost/(precision_cost+recall_cost)
fscore_bayes = 2*precision_bayes*recall_bayes/(precision_bayes+recall_bayes)

print('bayes with cost F score: : ',fscore_cost)
print('bayes F score: ',fscore_bayes)


# bayes with cost F score: :  0.6552534407519301
# bayes F score:  0.6606666666666667
```

Listing 4: 2d

2.5 e

```
avg_error_mdc = 0
avg_error_bayes = 0
for i in range(20):
    x0 = np.random.multivariate_normal(mu1, cov1,1000)
    x1 = np.random.multivariate_normal(mu2, cov2, 1000)

```

```
7       mdc_err = 0
8       bayes_err = 0
9
10      for i in x0:
11          if MDC(i)<0:
12              mdc_err+=1
13
14          if bayes(i)<0:
15              bayes_err +=1
16
17      for i in x1:
18          if MDC(i)>=0:
19              mdc_err+=1
20
21          if bayes(i)>=0:
22              bayes_err +=1
23
24      avg_error_bayes+=bayes_err/2000
25      avg_error_mdc+=mdc_err/2000
26
27  print('avg MDC error: ',avg_error_mdc/20)
28  print('avg bayes error: ',avg_error_bayes/20)
29
30  #avg MDC error:  0.015175000000000008
31  #avg bayes error:  0.012925000000000002
```

Listing 5: 2e

3 Automatic Saffron Cleaning Machine

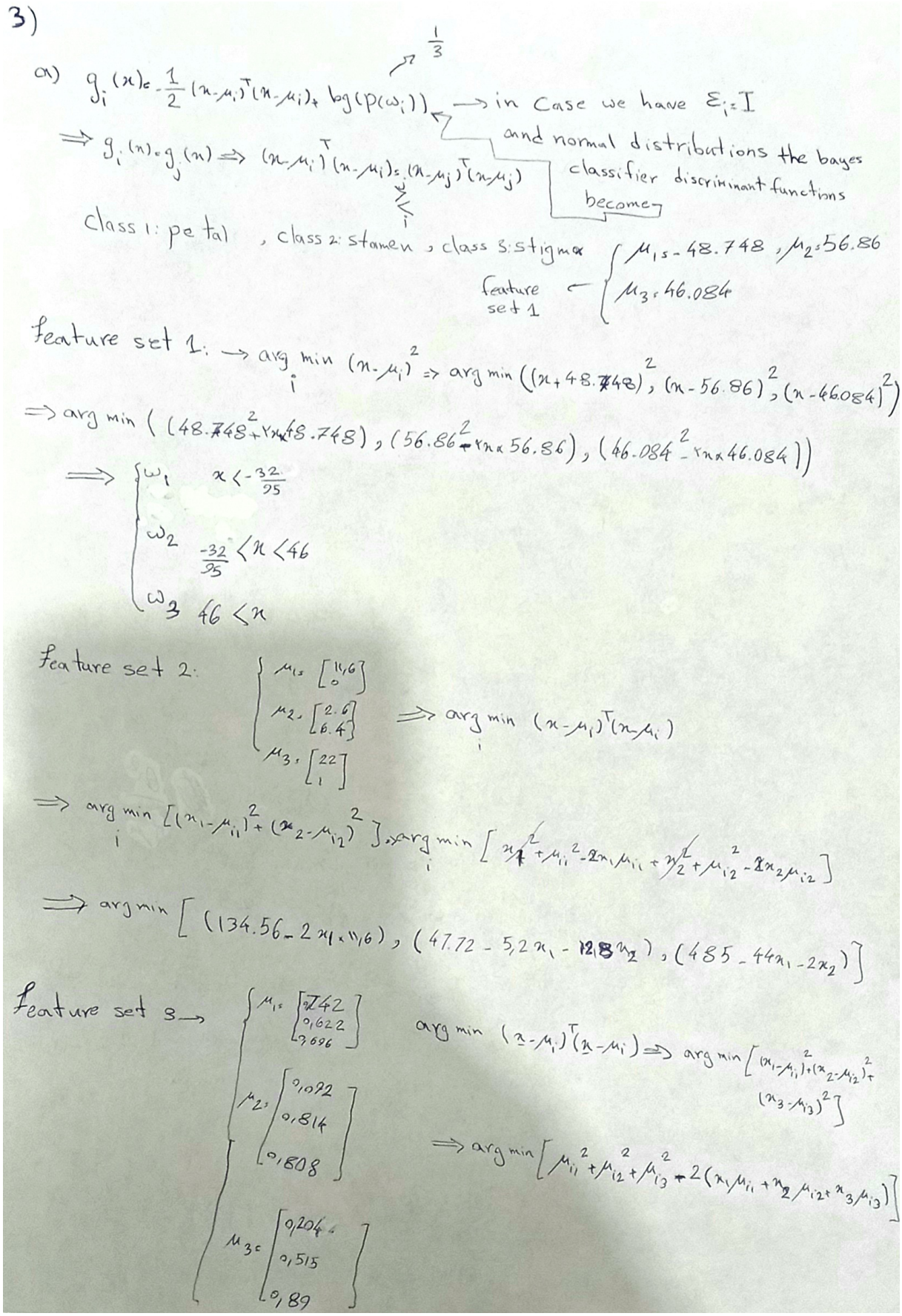


Figure 6: a

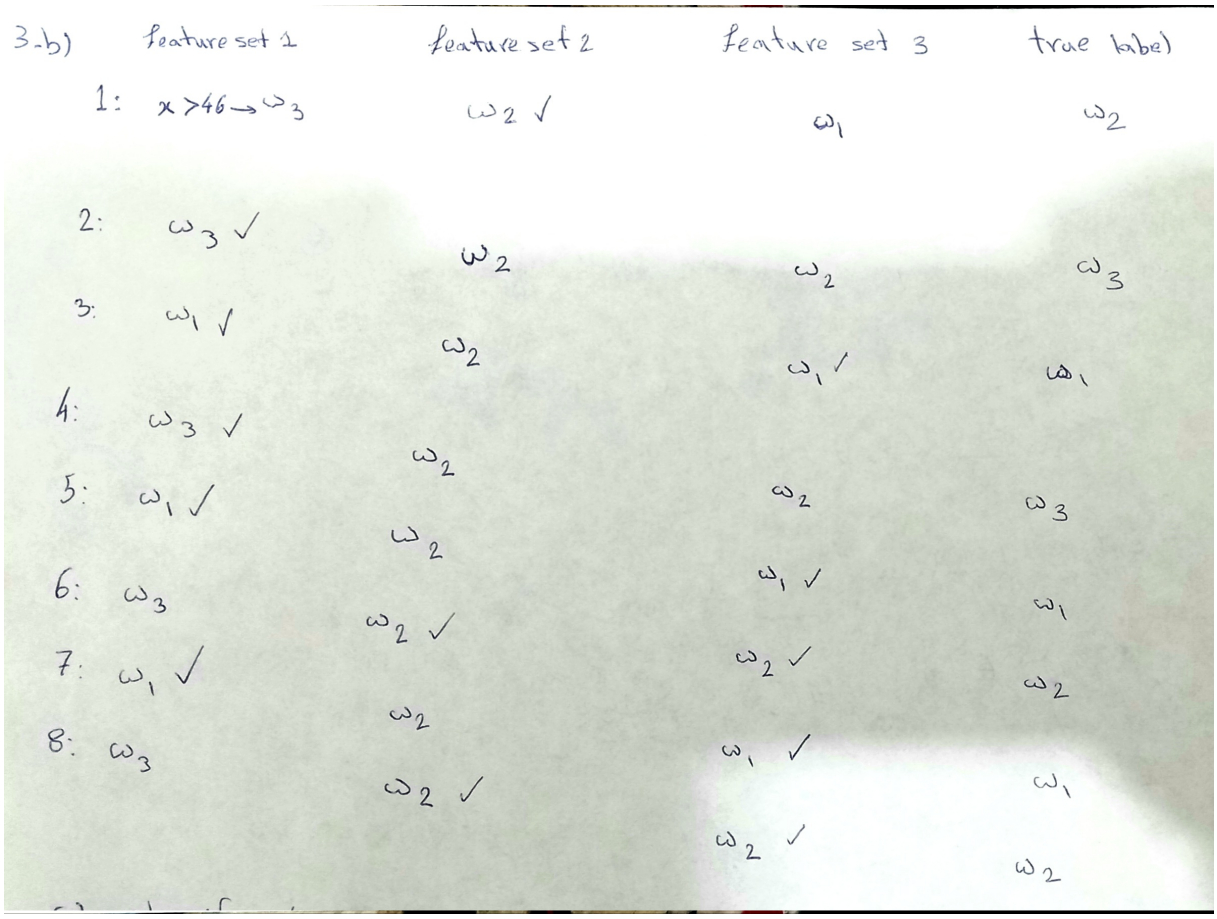

3-b)
feature set 1
feature set 2
feature set 3
true label
1: x > 46 → ω3
ω2 ✓
ω1
ω2
2: ω3 ✓
ω2
ω2
ω3
3: ω1 ✓
ω2
ω1 ✓
ω1
4: ω3 ✓
ω2
ω2
ω3
5: ω1 ✓
ω2
ω1 ✓
ω1
6: ω3
ω2 ✓
ω2 ✓
ω2
7: ω1 ✓
ω2
ω1 ✓
ω1
8: ω3
ω2 ✓
ω2 ✓
ω2

Figure 7: b

c) classifier 1:

true/predict	ω_1	ω_2	ω_3
ω_1	3	0	0
ω_2	0	0	3
ω_3	0	0	2

classifier 2:

true	true/Predict	Predict ω_1	Predict ω_2	Predict ω_3
	ω_1	0	3	0
	ω_2	0	3	0
	ω_3	0	2	0

classifier 3

true		Predict ω_1	Predict ω_2	Predict ω_3
	ω_1	3	0	0
	ω_2	1	2	0
	ω_3	0	2	0

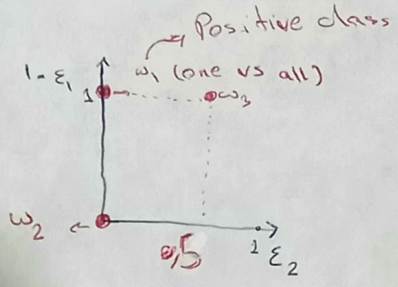
Figure 8: c

d) classifier 1 → error: $\frac{3}{8}$

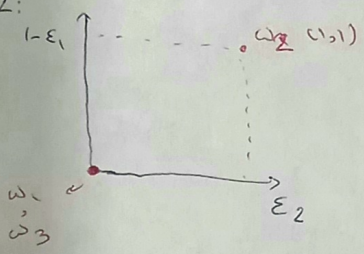
classifier 2 → error: $\frac{5}{8}$

classifier 3 → error: $\frac{3}{8}$

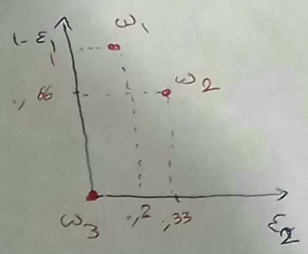
c) classifier 1:



classifier 2:



classifier 3:



f) we know that in bayes classifier if we have $\{\Sigma_i\}_i = I$ and normal distributions and equal priors the bayes discriminant function is equivalent to MDC classifier so we will have same results.

Figure 9: d-f

4 Dealing with Prediction Error in Bayes Decision Rule

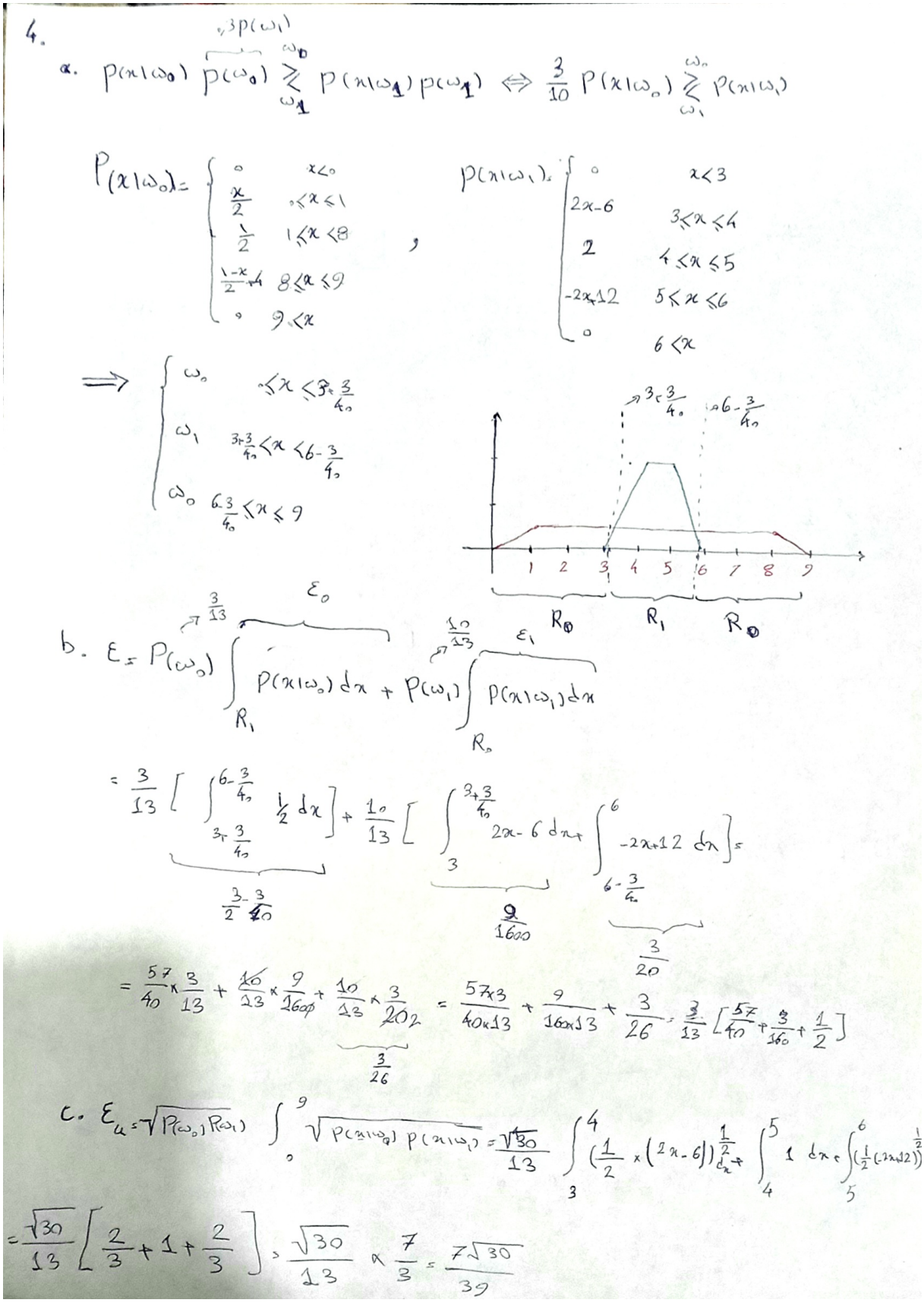


Figure 10: a-c

d. $\varepsilon_{min} = \underbrace{\left(\frac{3}{13}\right)^{s}\left(\frac{10}{13}\right)^{1-s}}_{A} \int_{-\infty}^{+\infty} p(x|\omega_0)^{s}\, p(x|\omega_1)^{1-s} = A\left[\int_{3}^{4}\left(\frac{1}{2}\right)^{s}(2x-6)^{1-s} + \int_{4}^{5}\left(\frac{1}{2}\right)^{s}(2)^{1-s} + \int_{5}^{6}\left(\frac{1}{2}\right)^{s}(-2x+12)^{1-s}\right]$

$= \left(\frac{3}{13}\right)^{s}\left(\frac{10}{13}\right)^{1-s}\left[\left(\frac{1}{2}\right)^{s}\int_{3}^{4}(2x-6)^{1-s} + 2^{1-2s} + 2^{-s}\int_{5}^{6}(-2x+12)^{1-s}\right]$

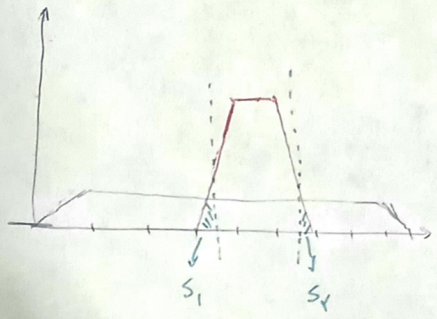
e. $\varepsilon_1 = \int_{R_0} P(x|\omega_1)\,dx = 0.05 = \int_{3}^{3+z} 2x-6\,dx + \int_{6-z}^{6} -2x+12 \Rightarrow \int_{3}^{z} 2x-6\,dx = \frac{1}{40}$

$\Rightarrow x^2 - 6x - 9 + 18 = \frac{1}{40} \Rightarrow z = 3 + \frac{1}{2\sqrt{10}} \Rightarrow R_1 = \left[3+\frac{1}{2\sqrt{10}}, 6-\frac{1}{2\sqrt{10}}\right]$, $R_0 = [0,9] - R_1$

$\varepsilon = P(\omega_0)\int_{R_1} p(x|\omega_0)\,dx + \underbrace{P(\omega_1)\times\frac{5}{100}}_{\frac{1}{26}} = \frac{1}{26} + \frac{3}{13}\int_{3+\frac{1}{2\sqrt{10}}}^{6-\frac{1}{2\sqrt{10}}} \frac{1}{2}\,dx = \frac{1}{26} + \frac{3}{13}\left[\frac{3}{2} - \frac{\sqrt{10}}{20}\right]$

$= \frac{1}{26} + \frac{9}{26} - \frac{3\sqrt{10}}{260} = \frac{1}{26}\left(10 - \frac{3\sqrt{10}}{10}\right)$

we know that for ε_1 to be equal to $\frac{5}{100}$ our decision region would be like this.



and $S_1 + S_2 = \frac{5}{100}$ so to minimize ε_0, S_1 and S_2 must be equal so $S_1 = S_2 = \frac{1}{40}$

Figure 11: d-e

5 Separating Messi from Football Field

5.1 a

```
def get_dct_feature(block):
    """make zigzag dct feature for a 8*8 normal block

    Args:
        block (numpy array 8*8)

    """
    zigzag = np.array([[0   , 1 ,  5 ,  6 , 14 , 15 , 27 , 28]

                       ,[2,   4 ,  7 , 13,  16 , 26,  29,  42]

                       ,[3,   8,  12 , 17 , 25 , 30 , 41 , 43]

                       ,[9 , 11 , 18 , 24 , 31 , 40 , 44 , 53]

                       ,[10 , 19 , 23 , 32 , 39 , 45 , 52,  54]

                       ,[20 , 22  ,33,  38  ,46,  51 , 55 , 60]

                       ,[21  ,34 , 37 , 47  ,50 , 56 , 59 , 61]

                       ,[35  ,36 , 48 , 49,  57,  58 , 62 , 63]])

    zigzag_vec = zigzag.reshape(64,)
    dct = cv2.dct(block) # DCT transformer
    dct_vec = np.abs(dct.reshape(64,))
    sec_larg_index = np.argpartition(dct_vec,-2)[-2] #finding the second largest abs
    value of DCTs
    return zigzag_vec[sec_larg_index]

#loading training images
img = cv2.imread('leo1.png',0)
mask_img = cv2.imread('leo1_mask.png',0)
#normalize image for dct
noraml_img = img/255
features = np.zeros((img.shape[0]//8 , img.shape[1]//8))

#calculating feature for each block
for i in range(img.shape[0]//8):
    for j in range(img.shape[1]//8):
        block = noraml_img[8*i:8*(i+1),8*j:8*(j+1)]
        features[i,j] = get_dct_feature(block)

#calculating priors
prior_messi = np.sum(mask_img/255)/(mask_img.shape[0]*mask_img.shape[1])
prior_field = 1 - prior_messi

print('prior messi = ' , prior_messi)
print('prior field = ' , prior_field)
#prior messi =  0.35684313725490197
#prior field =  0.6431568627450981
```

Listing 6: 5a

5.2 b

```
features_vec = features.reshape(img.shape[0]//8 * img.shape[1]//8,)
mask_features = np.zeros((img.shape[0]//8 , img.shape[1]//8))

#deciding if each block in training image belongs to feild or messi
for i in range(mask_img.shape[0]//8):
    for j in range(mask_img.shape[1]//8):
        block = mask_img[8*i:8*(i+1),8*j:8*(j+1)]/255
        if np.sum(block) > 32:
```

```
            mask_features[i,j] = 1
        else:
            mask_features[i,j] = 0

mask_features_vec = mask_features.reshape(mask_img.shape[0]//8 * mask_img.shape
    [1]//8,)

post_messi = []
post_field = []
#saving each feature to its class
for i in range(mask_img.shape[0]//8 * mask_img.shape[1]//8):
    if mask_features_vec[i] == 1:
        post_messi.append(features_vec[i])
    else:
        post_field.append(features_vec[i])

#plotting histograms
plt.hist(post_messi)
plt.savefig('messi likelihood.png')
plt.clf()
plt.hist(post_field)
plt.savefig('field likelihood.png')

#calculating likelihoods
counter_messi = collections.Counter(post_messi)

counter_field = collections.Counter(post_field)

for i in counter_field.keys():
    counter_field[i]/= len(post_field)

for i in counter_messi.keys():
    counter_messi[i]/= len(post_messi)

for i in range(64):
    if i not in counter_field.keys():
        counter_field[i] = 0

    if i not in counter_messi.keys():
        counter_messi[i] = 0
```

Listing 7: 5b

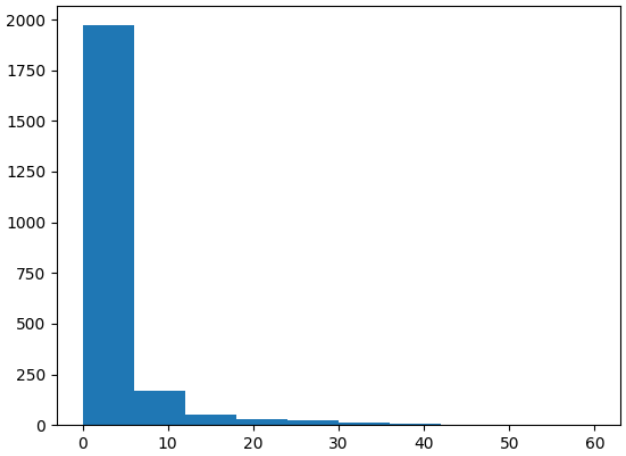


Figure 12: messi likelihood histogram

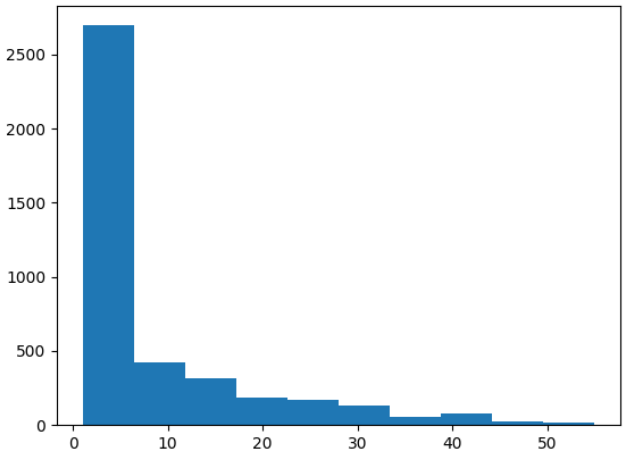


Figure 13: field likelihood histogram

5.3 c

```
#loading test image
test_img = cv2.imread('leo2.png',0)
normal_test = test_img / 255

test_features = np.zeros((test_img.shape[0]//8 , test_img.shape[1]//8))
# calculating the X feature for each block
for i in range(test_img.shape[0]//8):
    for j in range(test_img.shape[1]//8):
        block = normal_test[8*i:8*(i+1),8*j:8*(j+1)]
        test_features[i,j] =get_dct_feature(block)
```

Listing 8: 5c

5.4 d

```
#loading test image
test_img = cv2.imread('leo2.png',0)
normal_test = test_img / 255

test_features = np.zeros((test_img.shape[0]//8 , test_img.shape[1]//8))
# calculating the X feature for each block
for i in range(test_img.shape[0]//8):
    for j in range(test_img.shape[1]//8):
        block = normal_test[8*i:8*(i+1),8*j:8*(j+1)]
        test_features[i,j] =get_dct_feature(block)
        x = test_features[i,j]
        if counter_messi[x]*prior_messi>counter_field[x]*prior_field:
            #test_img[8*i:8*(i+1),8*j:8*(j+1)]=np.ones((8,8))*255
            pass
        else:
            test_img[8*i:8*(i+1),8*j:8*(j+1)]=np.zeros((8,8))

cv2.imwrite('result.png',test_img)
```

Listing 9: 5d

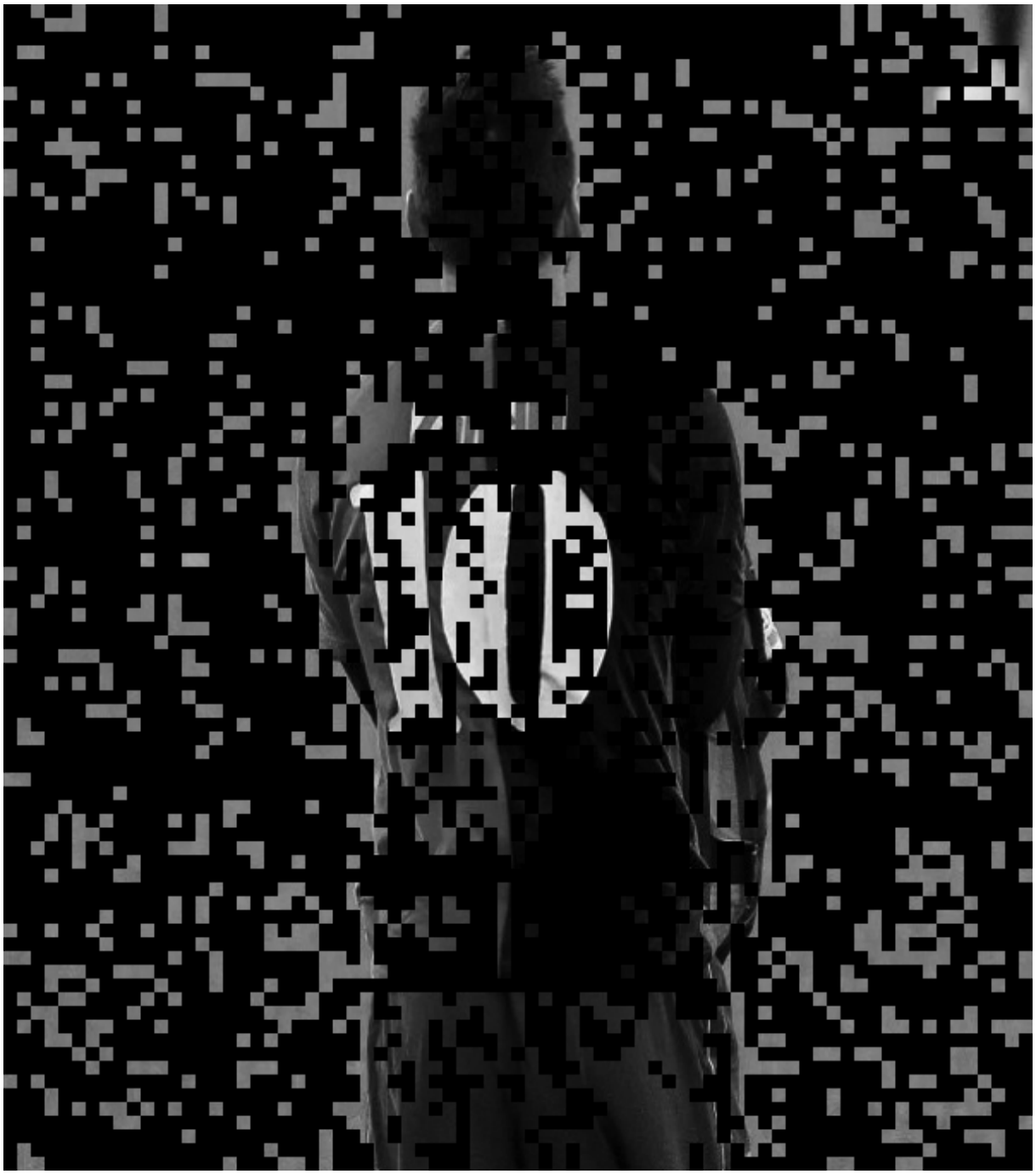


Figure 14: result

5.5 e

```
1
2  #loading test image
3  test_img = cv2.imread('leo2.png',0)
4  normal_test = test_img / 255
5  test_mask = cv2.imread('leo2_mask.png',0)
6
7  test_features = np.zeros((test_img.shape[0]//8 , test_img.shape[1]//8))
8  error = 0
9  # calculating the X feature for each block
10 for i in range(test_img.shape[0]//8):
11     for j in range(test_img.shape[1]//8):
```

```
            block = normal_test[8*i:8*(i+1),8*j:8*(j+1)]
            test_features[i,j] =get_dct_feature(block)
            x = test_features[i,j]
            if counter_messi[x]*prior_messi>counter_field[x]*prior_field:
                if np.sum(test_mask[8*i:8*(i+1),8*j:8*(j+1)]/255)<33:
                    error+=1
            else:
                if np.sum(test_mask[8*i:8*(i+1),8*j:8*(j+1)]/255)>32:
                    error+=1

error = error / ((test_mask.shape[0]//8) * (test_mask.shape[1]// 8))

print("prob of error: ", error)

#prob of error:  0.2912941176470588
```

Listing 10: 5e

6 Maximum Likelihood Approach for Parameter Estimation

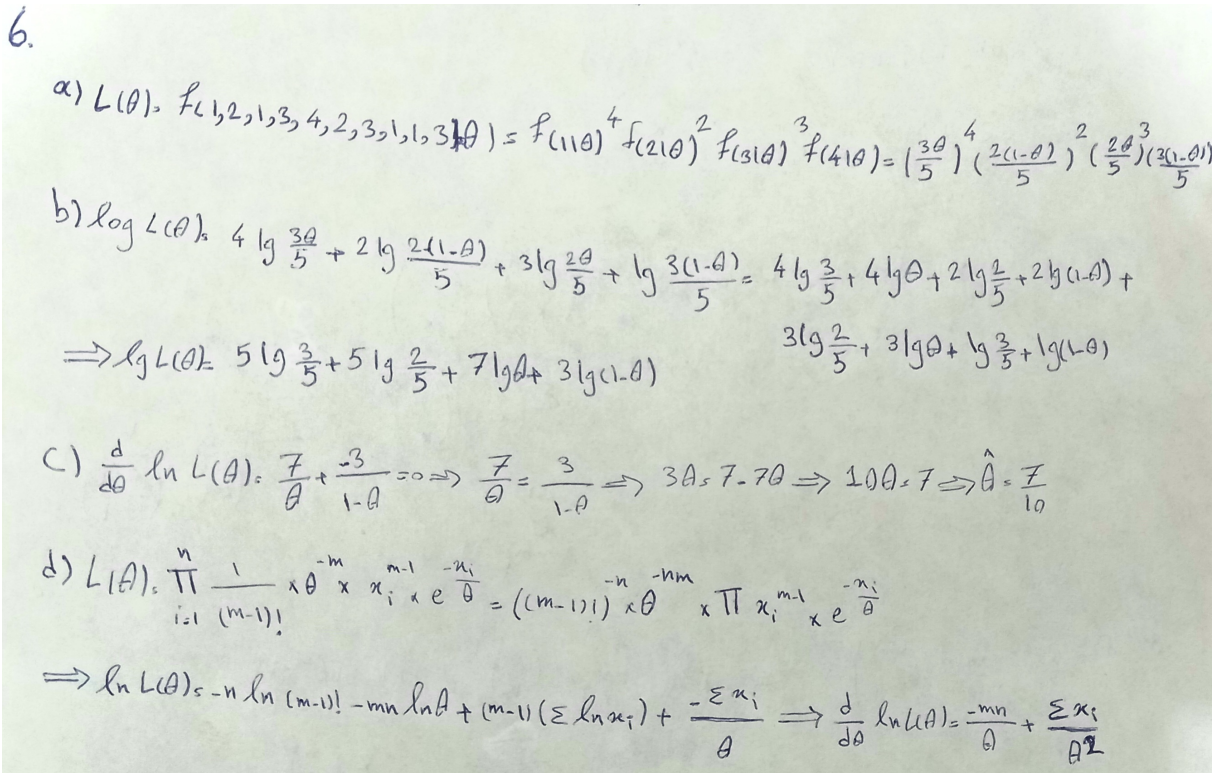


Figure 15: a-d

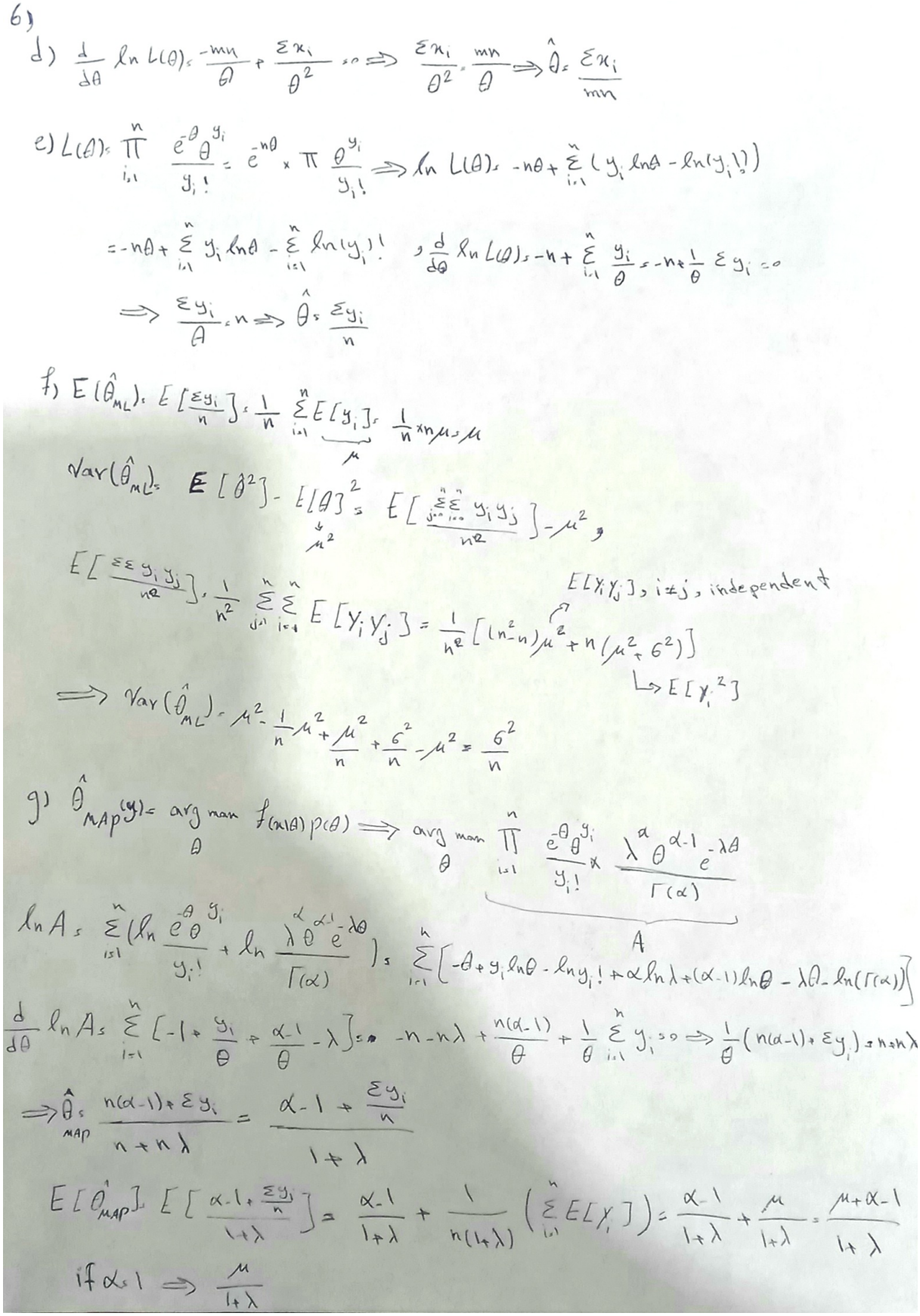

6)
d) $\frac{d}{d\theta} \ln L(\theta) = \frac{-mn}{\theta} + \frac{\sum x_i}{\theta^2} = 0 \Rightarrow \frac{\sum x_i}{\theta^2} = \frac{mn}{\theta} \Rightarrow \hat{\theta} = \frac{\sum x_i}{mn}$
e) $L(\theta) = \prod_{i=1}^{n} \frac{e^{-\theta}\theta^{y_i}}{y_i!} = e^{-n\theta} \times \pi \frac{\theta^{y_i}}{y_i!} \Rightarrow \ln L(\theta) = -n\theta + \sum_{i=1}^{n} (y_i \ln\theta - \ln(y_i!))$
$= -n\theta + \sum_{i=1}^{n} y_i \ln\theta - \sum_{i=1}^{n} \ln(y_i!)$, $\frac{d}{d\theta} \ln L(\theta) = -n + \sum_{i=1}^{n} \frac{y_i}{\theta} = -n + \frac{1}{\theta} \sum y_i = 0$
$\Rightarrow \frac{\sum y_i}{\theta} = n \Rightarrow \hat{\theta} = \frac{\sum y_i}{n}$
f) $E(\hat{\theta}_{ML}) = E\left[\frac{\sum y_i}{n}\right] = \frac{1}{n} \sum_{i=1}^{n} E[y_i] = \frac{1}{n} \times n\mu = \mu$
$Var(\hat{\theta}_{ML}) = E[\hat{\theta}^2] - E[\hat{\theta}]^2 = E\left[\frac{\sum\sum y_i y_j}{n^2}\right] - \mu^2$,
$E\left[\frac{\sum\sum y_i y_j}{n^2}\right] = \frac{1}{n^2} \sum_{j=1}^{n} \sum_{i=1}^{n} E[y_i y_j] = \frac{1}{n^2}\left[(n^2-n)\mu^2 + n(\mu^2+\sigma^2)\right]$
$E[X_i Y_j]$, $i \neq j$, independent
$\hookrightarrow E[X_i^2]$
$\Rightarrow Var(\hat{\theta}_{ML}) = \mu^2 - \frac{1}{n}\mu^2 + \frac{\mu^2}{n} + \frac{\sigma^2}{n} - \mu^2 = \frac{\sigma^2}{n}$
g) $\hat{\theta}_{MAP}(y) = \arg\max_{\theta} f(x|\theta) p(\theta) \Rightarrow \arg\max_{\theta} \prod_{i=1}^{n} \frac{e^{-\theta}\theta^{y_i}}{y_i!} \times \frac{\lambda^{\alpha}\theta^{\alpha-1}e^{-\lambda\theta}}{\Gamma(\alpha)}$
A
$\ln A = \sum_{i=1}^{n} \left(\ln \frac{e^{-\theta}\theta^{y_i}}{y_i!} + \ln \frac{\lambda^{\alpha}\theta^{\alpha-1}e^{-\lambda\theta}}{\Gamma(\alpha)}\right) = \sum_{i=1}^{n} \left[-\theta + y_i \ln\theta - \ln y_i! + \alpha\ln\lambda + (\alpha-1)\ln\theta - \lambda\theta - \ln(\Gamma(\alpha))\right]$
$\frac{d}{d\theta} \ln A = \sum_{i=1}^{n} \left[-1 + \frac{y_i}{\theta} + \frac{\alpha-1}{\theta} - \lambda\right] = 0 \Rightarrow -n - n\lambda + \frac{n(\alpha-1)}{\theta} + \frac{1}{\theta} \sum_{i=1}^{n} y_i = 0 \Rightarrow \frac{1}{\theta}(n(\alpha-1) + \sum y_i) = n + n\lambda$
$\Rightarrow \hat{\theta}_{MAP} = \frac{n(\alpha-1) + \sum y_i}{n + n\lambda} = \frac{\alpha - 1 + \frac{\sum y_i}{n}}{1+\lambda}$
$E[\hat{\theta}_{MAP}] = E\left[\frac{\alpha-1+\frac{\sum y_i}{n}}{1+\lambda}\right] = \frac{\alpha-1}{1+\lambda} + \frac{1}{n(1+\lambda)}\left(\sum_{i=1}^{n} E[X_i]\right) = \frac{\alpha-1}{1+\lambda} + \frac{\mu}{1+\lambda} = \frac{\mu+\alpha-1}{1+\lambda}$
if $\alpha = 1 \Rightarrow \frac{\mu}{1+\lambda}$

Figure 16: d-g

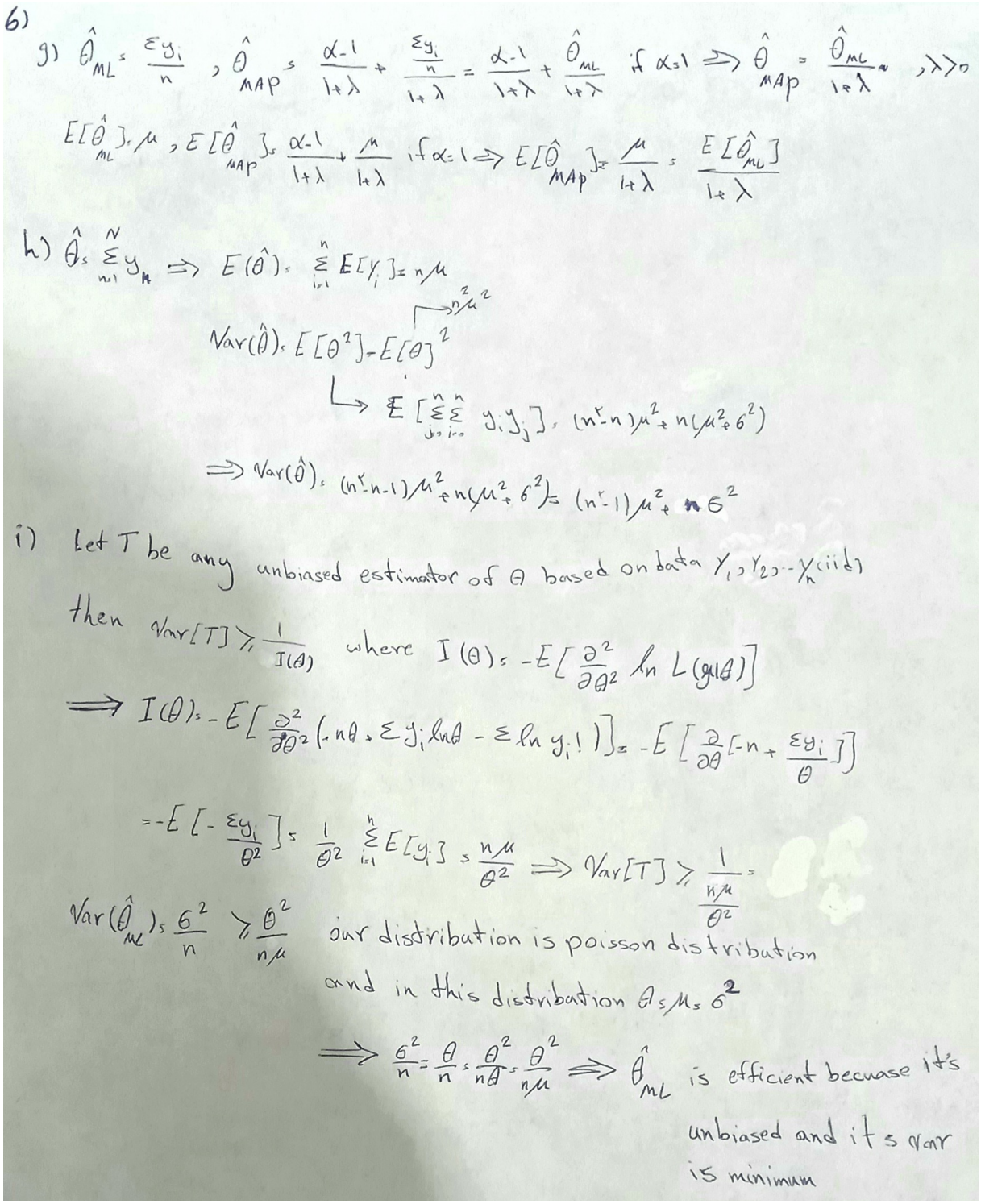

6)
g) $\hat{\theta}_{ML} = \frac{\sum y_i}{n}$, $\hat{\theta}_{MAP} = \frac{\alpha-1}{1+\lambda} + \frac{\sum y_i}{n(1+\lambda)} = \frac{\alpha-1}{1+\lambda} + \frac{\hat{\theta}_{ML}}{1+\lambda}$ if $\alpha=1 \Rightarrow \hat{\theta}_{MAP} = \frac{\hat{\theta}_{ML}}{1+\lambda}$, $\lambda>0$
$E[\hat{\theta}_{ML}] = \mu$, $E[\hat{\theta}_{MAP}] = \frac{\alpha-1}{1+\lambda} + \frac{n\mu}{1+\lambda}$ if $\alpha=1 \Rightarrow E[\hat{\theta}_{MAP}] = \frac{\mu}{1+\lambda} = \frac{E[\hat{\theta}_{ML}]}{1+\lambda}$
h) $\hat{\theta} = \sum_{n=1}^{N} y_n \Rightarrow E(\hat{\theta}) = \sum_{i=1}^{n} E[y_i] = n\mu$
$n^2\mu^2$
$Var(\hat{\theta}) = E[\theta^2] - E[\theta]^2$
$E[\sum_{j=1}^{n}\sum_{i=1}^{n} y_i y_j] = (n^2-n)\mu^2 + n(\mu^2+\sigma^2)$
$\Rightarrow Var(\hat{\theta}) = (n^2-n-1)\mu^2 + n(\mu^2+\sigma^2) = (n^2-1)\mu^2 + n\sigma^2$
i) Let T be any unbiased estimator of θ based on data $Y_1, Y_2, \dots Y_n$ (i.i.d)
then $Var[T] \geq \frac{1}{I(\theta)}$ where $I(\theta) = -E[\frac{\partial^2}{\partial\theta^2} \ln L(y|\theta)]$
$\Rightarrow I(\theta) = -E[\frac{\partial^2}{\partial\theta^2}(-n\theta + \sum y_i \ln\theta - \sum \ln y_i!)] = -E[\frac{\partial}{\partial\theta}[-n + \frac{\sum y_i}{\theta}]]$
$= -E[-\frac{\sum y_i}{\theta^2}] = \frac{1}{\theta^2}\sum_{i=1}^{n} E[y_i] = \frac{n\mu}{\theta^2} \Rightarrow Var[T] \geq \frac{1}{\frac{n\mu}{\theta^2}}$
$Var(\hat{\theta}_{ML}) = \frac{\sigma^2}{n} \geq \frac{\theta^2}{n\mu}$ our distribution is poisson distribution
and in this distribution $\theta = \mu = \sigma^2$
$\Rightarrow \frac{\sigma^2}{n} = \frac{\theta}{n} = \frac{\theta^2}{n\theta} = \frac{\theta^2}{n\mu} \Rightarrow \hat{\theta}_{ML}$ is efficient because it's
unbiased and it's var
is minimum

Figure 17: g-i

7 Here's Why Hitler Hated MLE So Bad

7.

a) num of all possible outcomes $\rightarrow \binom{n}{k}$

num of cases when highest serial number is m $\rightarrow \binom{m-1}{k-1}$ because when the largest serial number is m we should Pick other $k-1$ serial numbers from 1 to $m-1$

so $P(m|n,k) = \dfrac{\binom{m-1}{k-1}}{\binom{n}{k}}$

b) $P(n|m,k) = \dfrac{P(m|n,k)\,P(n|k)}{P(m|k)}$ (1)

$P(n|k) = \dfrac{1}{\Omega - k}$ $\rightarrow$ because $k \le n < \Omega$ and n is uniformly distributed

$P(m|k) = P(m|k) \times 1 = P(m|k) \times \sum_{n=0}^{\Omega-1} P(n|m,k) \overset{(1)}{=} P(m|k) \sum_{n=m}^{\Omega-1} \dfrac{P(m|n,k)P(n|k)}{P(m|k)}$

$= \sum_{n=m}^{\Omega-1} P(m|n,k)\underbrace{P(n|k)}_{\frac{1}{\Omega-k}} = \dfrac{\sum_{n=m}^{\Omega-1} P(m|n,k)}{\Omega - k}$

$\Longrightarrow P(n|m,k) = \dfrac{P(m|n,k) \times \frac{1}{\Omega-k}}{\frac{1}{\Omega-k}\sum_{n=m}^{\Omega-1} P(m|n,k)} = \dfrac{\binom{m-1}{k-1}\binom{n}{k}^{-1}}{\sum_{n=m}^{\Omega-1}\binom{m-1}{k-1}\binom{n}{k}^{-1}} = \dfrac{\binom{n}{k}^{-1}}{\underbrace{\sum_{n=m}^{\Omega-1}\frac{1}{\binom{n}{k}}}_{\text{using Hint}}} = \dfrac{\binom{n}{k}^{-1}}{\frac{k}{k-1}\left[\frac{1}{\binom{m-1}{k-1}} - \frac{1}{\binom{\Omega-1}{k-1}}\right]}$

if $\Omega \to \infty$ then $\binom{\Omega-1}{k-1} \to \infty \Longrightarrow P(n|m,k) = \dfrac{\binom{n}{k}^{-1}}{\frac{k}{\binom{m-1}{k-1}(k-1)}} = \dfrac{(k-1)\binom{m-1}{k-1}}{k\binom{n}{k}}$

Figure 18: a-b

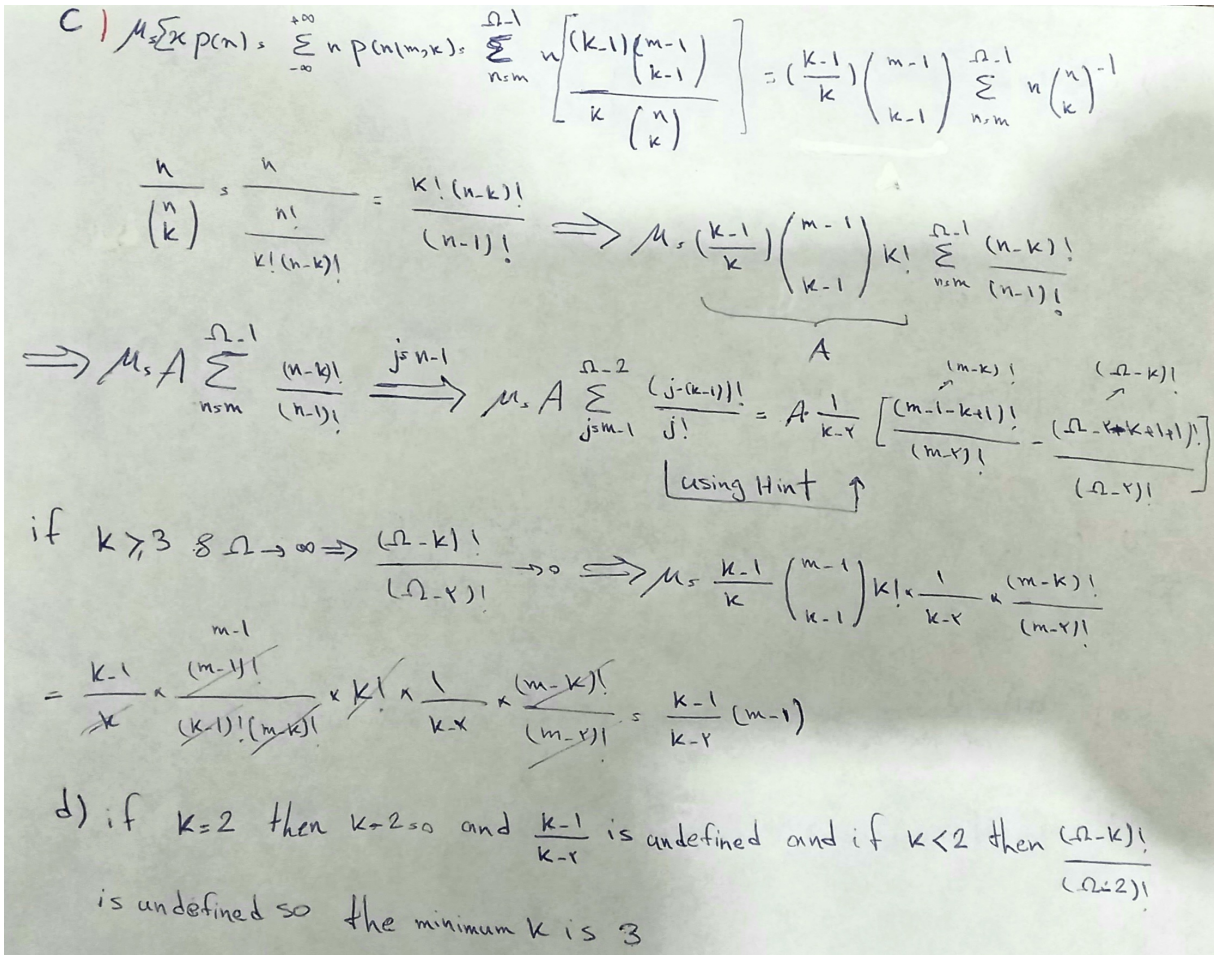


Figure 19: c-d

8 Further Study: When Will Civilization End?

9 Some Explanatory Questions

9.1 a

when the prior probabilities are equal and two classes have normal distributions with the same covariance matrix. and the covariance matrix has to be equal to the identity matrix.

9.2 b

in MDC when we want to choose class representative for example mean of each class we have to calculate it if we dont have it and its training phase but if we have class means we dont need to calculate. similarly in minimum error classifier if we dont have distributions or parameters of distribution we have to estimate them using data and this could be training phase but if we have distributions we dont need to this phase.

9.3 c

the Bayes error is the best error that we can achieve because we have the exact distributions and probabilities and we have an infinitely large dataset and we know in this situation Bayes error is optimum.

9.4 d

yes we can use one vs all method or we can plot ROC curve for each pair of classes.

9.5 e

yes we can assume a distribution for target and use the bayes rule for calculating targets

9.6 f

yes it is. because we have a decision boundary in bayes decision rule and its fixed.

9.7 g

if in MAP we have uniform prior then the MAP and MLE are the same. in MAP we use more information for estimation so in general its better than MLE but if we dont have a correct prior its better to use MLE.

9.8 h

in this method we add -lambdaP(theta) to the function that we want to maximize. when we have a small amount of data its better to apply penalized MLE