Assignment II

Date :

*Problem 1:

⇒ According to the question, We will first execute it's mean,

$$E[x] = \int_0^\infty x \cdot \text{Gamma}(x|a,b)\,dx$$

$$= \int_0^\infty x \frac{b^a}{\Gamma(a)} x^{a-1} e^{-bx}\,dx$$

$$= \int_0^\infty x \frac{b^a}{\Gamma(a)} x^{a-1} e^{-bx}\,dx$$

$$= \frac{b^a}{\Gamma(a)} \int_0^\infty x \cdot x^{a-1} \cdot e^{-bx}\,dx$$

$$= \frac{b^a}{\Gamma(a)} \int_0^\infty x^a e^{-bx}\,dx$$

By the property we know that,

$$\int_0^\infty x^{a-1} e^{bx}\,dx = \frac{\Gamma(a)}{b^a}$$

So, $$\int_0^\infty x^a e^{-bx}\,dx = \frac{\Gamma(a+1)}{b^{a+1}}$$

$$E[x] = \frac{b^a}{\Gamma(a)} \frac{\Gamma(a+1)}{b^{a+1}},$$

By the property of $\Gamma(a+1) = a\Gamma(a)$

Date :

$$E[x] = \frac{b^a}{\Gamma(a)} \frac{a\Gamma(a)}{b^a b}$$

$$= a/b$$

Now we need to calculate the Variance,

$$Var[x] = E[x^2] - (E[x])^2$$

So, $E[x^2] = \int_0^\infty x^2 \frac{b^a}{\Gamma(a)} x^{a-1} e^{-bx} dx$

$$= \frac{b^a}{\Gamma(a)} \int_0^\infty x^2 . x^{a-1} e^{-bx} dx$$

$$= \frac{b^a}{\Gamma(a)} \int_0^\infty x^2 . x^{a-1} e^{-bx} dx$$

$$= \frac{b^a}{\Gamma(a)} \int_0^\infty x^{a+1} e^{-bx} dx$$

By Using the Same properties we used for mean,

$$E[x^2] = \frac{b^a}{\Gamma(a)} \frac{\Gamma(a+2)}{b^{a+2}}$$

$$= \frac{b^a}{\Gamma(a)} \frac{\Gamma(a+2)}{b^{a+2}}$$

Date :

$$= \frac{b^a}{\Gamma(a)} \frac{(a+1)a\Gamma(a)}{b^2 b^a}$$

$$= \frac{a(a+1)}{b^2}$$

$$\therefore Var[x] = \frac{a(a+1)}{b^2} - \frac{a^2}{b^2}$$

$$= \frac{a}{b^2}$$

Date :

*Problem 2: Given that,

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1}+D^{-1}CMBD^{-1} \end{pmatrix}$$

$$M = (A-BD^{-1}C)^{-1}$$

We have to prove, $Y^{-1} = X$

Also, $XY = I$ [I = Identity Matrix]

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1}+D^{-1}CMBD^{-1} \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

$$M_{11} = AM - BD^{-1}CM = M(A-BD^{-1}C) = (A-BD^{-1}C)^{-1} \cdot (A-BD^{-1}C) = I$$

$$M_{12} = -AMBD^{-1} + BD^{-1} + BD^{-1}CMBD^{-1}$$
$$= BD^{-1}(-AM + I + CMBD^{-1})$$

So, therefore,

$$-AM + CMBD^{-1} = -I, \quad BD^{-1}(I-I) = 0$$

$$M_{21} = CM - DD^{-1}CM = CM - CM = 0$$

$$M_{22} = -CMBD^{-1} + DD^{-1} + DD^{-1}CMBD^{-1}$$
$$= -CMBD^{-1} + I + CMBD^{-1}$$
$$= I$$

Date :

Hence,

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1}+D^{-1}CMBD^{-1} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$$

Also, $$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1}+D^{-1}CMBD^{-1} \end{pmatrix}$$

Date :..

*Problem 3:

$$\Rightarrow R = \begin{pmatrix} \Lambda + A^T L A & -A^T L \\ -LA & L \end{pmatrix}$$

By using the property,

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{pmatrix}$$

$$M = (\Lambda + A^T L A - (A^T L L^{-1} - LA))^{-1}$$

$$= (\Lambda + A^T L A - (-A^T - LA)^{-1}$$

$$= \Lambda^{-1}$$

$$\therefore R^{-1} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

$M_{11} = \Lambda^{-1}$

$M_{12} = -\Lambda^{-1} \cdot A^T L L^{-1} = \Lambda^{-1} A^T$

$M_{21} = (-L^{-1})(-LA)(\Lambda^{-1}) = A\Lambda^{-1}$

$M_{22} = L^{-1} + (L^{-1})(-LA\Lambda^{-1})(-A^T) L L^{-1}$

$= L^{-1} A \Lambda^{-1} A^T$

Hence, $\operatorname{cov}[z] = R^{-1} = \begin{pmatrix} \Lambda^{-1} & \Lambda^{-1} A^T \\ A\Lambda^{-1} & L^{-1} A \Lambda^{-1} A^T \end{pmatrix}$

Date :

*Problem 4: Given that,

$\Rightarrow p(x) = \mathcal{N}(x|\mu_x, \Sigma_x)$

$p(z) = \mathcal{N}(z|\mu_z, \Sigma_z)$

$y = x + z.$

We know that, $E[x] = x$

$cov[x] = E[x^2] - (E[x])^2 = 0$

therefore,

$\mu_{y|x} = E[x] + E[z]$

$= x + \mu_z$

$\Sigma_{y|x} = cov[x] + cov[z] = \Sigma_z$

Also, $p(y|x) = \mathcal{N}(y|x + \mu_z, \Sigma_z)$

By comparing the equation,

$p(x) = \mathcal{N}(x|\mu, \Lambda^{-1})$

$p(y|x) = \mathcal{N}(y|Ax + b, L^{-1})$

$p(y) = \mathcal{N}(y + A\mu + b, L^{-1} + A\Lambda^{-1}A^T)$

$\mu = \mu_x$

$A = I$

$b = \mu_z$

$\Lambda^{-1} = \Sigma_x$

$L^{-1} = \Sigma_z$

$p(y) = \mathcal{N}(y|\mu_x + \mu_z, \Sigma_z + A\Sigma_x A^T)$

$= \mathcal{N}(y|\mu_x + \mu_z, \Sigma_z + \Sigma_x)$

Assignment 2

Date ....

*5

→ In this problem, we are requested to approach the problem from a different angle in this question. We need to write the joint distribution $p(x, y)$ and then embed over to obtain the marginal distribution $p(y)$.

Lets start by writing the quadratic form of $p(x, y)$ in exponential form:

$$-\frac{1}{2}(x-\mu)^T \Lambda (x-\mu) - \frac{1}{2}(y-Ax-b)^T L (y-Ax-b)$$

Those terms involving x are retrieved:

$$= -\frac{1}{2} x^T (\Lambda + A^T L A) x + x^T [\Lambda\mu + A^T L (y-b)] + \text{constant}$$

$$= -\frac{1}{2} (x-m)^T (\Lambda + A^T L A)(x-m) + \frac{1}{2} m^T (\Lambda + A^T L A) m + \text{constant}$$

Here's what we have come up with:

$$m = (\Lambda + A^T L A)^{-1} [\Lambda\mu + A^T L (y-b)]$$

When we integrate over x, we can observe that the first term dissappeared to a constant, then we extract the remaining terms that include y.

Date :

We can get the following parts:

$$= \frac{1}{2} y^T [L - LA(\Lambda + A^T LA)^{-1} A^T L] y + y^T \{[L - LA(\Lambda + A^T LA^{-1} A^T L] b + LA(\Lambda + A^T LA)^{-1} \Lambda \mu\}$$

By taking the inverse of $y^T y$ coefficient, we can get the covariance matrix cov[y]

$$L - LA(\Lambda + A^T LA)^{-1} A^T L$$

By utilizing Woodbury inversion formula:

$$(X + YZU) = X^{-1} - X^{-1} Y (Z^{-1} + U X^{-1} Y)^{-1} U X^{-1}$$

where, $X^{-1} = L$, $Y = A$, $Z^{-1} = \Lambda$, $U = A^T$

so that, we can perform

$$\text{cov}[y] = (L^{-1} + A \Lambda^{-1} A^T)$$

Coefficient y^T must be equal to $E[y] (\text{cov}[y])^{-1}$

$$E[y] (L^{-1} + A \Lambda^{-1} A^T)^{-1} = \{[L - LA(\Lambda + A^T LA)^{-1} A^T L] b + LA(\Lambda + A^T LA)^{-1} \Lambda \mu\}$$

$$E[y] = (L^{-1} + A \Lambda^{-1} A^T) \{[L - LA(\Lambda + A^T LA)^{-1} A^T L] b + LA(\Lambda + A^T LA)^{-1} \Lambda \mu\}$$

$$E[y] = (L^{-1} + A \Lambda^{-1} A^T) \{(L^{-1} + A \Lambda^{-1} A^T)^{-1} b + LA(\Lambda + A^T LA)^{-1} \Lambda \mu\}$$

$$E[y] = \{b + (L^{-1} + A \Lambda^{-1} A^T) LA (\Lambda + A^T LA)^{-1} \Lambda \mu\}$$

Date :

By utilizing Woodbury inversion method, we can perform

$$(\Lambda+A^TLA)^{-1} = \Lambda^{-1} - \Lambda^{-1}A^T(L^{-1}+A\Lambda^{-1}A^T)^{-1}A\Lambda^{-1}$$

$$E[y] = b + (L^{-1}+A\Lambda^{-1}A^T)LA(\Lambda^{-1} - \Lambda^{-1}A^T)(L^{-1}+A\Lambda^{-1}A^T)^{-1}A\Lambda^{-1}$$

$$= b + (L^{-1}+A\Lambda^{-1}A^T)LA\Lambda^{-1} - (L^{-1}+A\Lambda^{-1}A^T)LA\Lambda^{-1}A^T(L^{-1}+A\Lambda^{-1}A^T)^{-1}A\Lambda^{-1}$$

$$= b + (L^{-1}+A\Lambda^{-1}A^T)LA\Lambda^{-1} - LA\Lambda^{-1}A^T\Lambda^{-1}$$

$$= b + L^{-1}LA\Lambda^{-1} + A\Lambda^{-1}A^TLA\Lambda^{-1} - A\Lambda^{-1}A^TLA\Lambda^{-1}$$

$$= b + A\Lambda^{-1}$$

$$\therefore E[y] = b + A\Lambda^{-1}\Lambda\mu$$

$$= A\mu + b$$

In [18]:

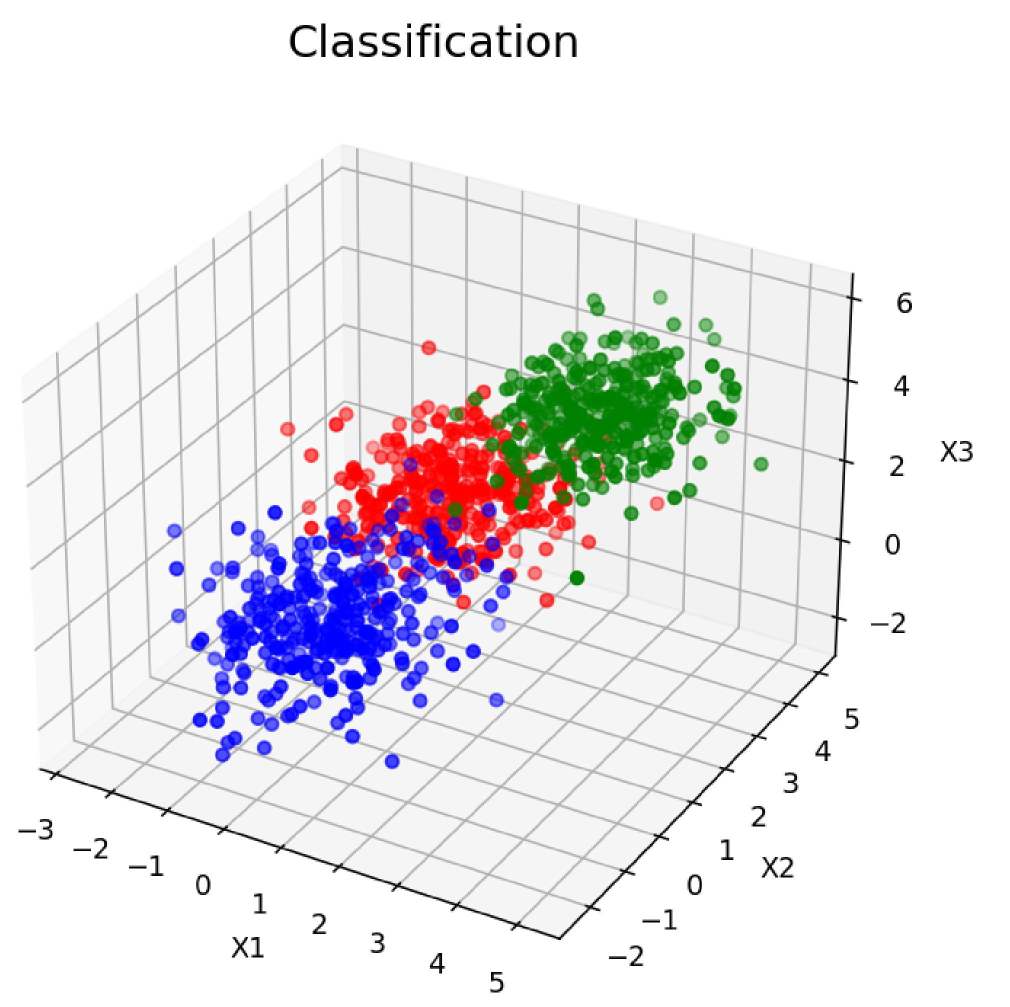
```python
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
```

In [19]:

```python
N = 1000
Pw0 = Pw1 = Pw2 = 1/3       # A priori probabilities for each class are equal

m1 = np.array([0, 0, 0])
m2 = np.array([1, 2, 2])
m3 = np.array([3, 3, 4])


#S = S0 = S1 = S2 = 0.8*np.eye(3)

S1 = np.array([[0.8,0.2,0.1],
               [0.2,0.8,0.2],
               [0.1,0.2,0.8]])
S2 = np.array([[0.6,0.01,0.01],
               [0.01,0.8,0.01],
               [0.01,0.01,0.6]])
S3 = np.array([[0.6,0.1,0.1],
               [0.1,0.6,0.1],
               [0.1,0.1,0.6]])

# Creating a training set
Xtr_w0 = np.random.multivariate_normal(m1, S1, 333)  # a set of vectors for class 0
ytr_w0 = 0*np.ones((333, 1))                         # class 0's labels

Xtr_w1 = np.random.multivariate_normal(m2, S2, 333)  # a set of vectors for class 1
ytr_w1 = 1*np.ones((333, 1))                         # class 1's labels

Xtr_w2 = np.random.multivariate_normal(m3, S3, 333)  # a set of vectors for class 2
ytr_w2 = 2*np.ones((333, 1))                         # class 2's labels

# Data and labels are collected in a single set.
Xtr = np.concatenate((Xtr_w0, Xtr_w1, Xtr_w2), axis = 0)
ytr = np.concatenate((ytr_w0, ytr_w1, ytr_w2), axis = 0)

## creating a test set

Xte_w0 = np.random.multivariate_normal(m1, S1, 333)    # a set of vectors for class 0
yte_w0 = 0*np.ones((333, 1))                          # class 0's labels

Xte_w1 = np.random.multivariate_normal(m2, S2, 333)    # a set of vectors for class 1
yte_w1 = 1*np.ones((333, 1))                          # class 1's labels

Xte_w2 = np.random.multivariate_normal(m3, S3, 333)    # a set of vectors for class 2
yte_w2 = 2*np.ones((333, 1))                          # class 2's labels

# Data and labels are collected in a single set.
Xte = np.concatenate((Xte_w0, Xte_w1, Xte_w2), axis = 0)
yte = np.concatenate((yte_w0, yte_w1, yte_w2), axis = 0)
```

In [20]:

```python
%matplotlib notebook
fig = plt.figure(figsize = (6,6))
ax  = fig.add_subplot(projection = "3d")

ax.scatter(Xtr_w0[:,0], Xtr_w0[:,1], Xtr_w0[:,2], marker = "o", color = "b", label = "Class-0")
ax.scatter(Xtr_w1[:,0], Xtr_w1[:,1], Xtr_w1[:,2], marker = "o", color = "r", label = "Class-1")
ax.scatter(Xtr_w2[:,0], Xtr_w2[:,1], Xtr_w2[:,2], marker = "o", color = "g", label = "Class-2")
ax.set_xlabel('X1')
ax.set_ylabel('X2')
ax.set_zlabel('X3')
plt.title(r'Classification', fontsize=16)
plt.show()
```



Part i: Estimates based on ML

In [21]:

```python
m1_hat = (1.0/(N/3))*np.sum(Xtr_w0, axis = 0)
S0_hat = (1.0/(N/3))*np.dot((Xtr_w0-m1_hat).T,(Xtr_w0-m1_hat))

m2_hat = (1.0/(N/3))*np.sum(Xtr_w1, axis = 0)
S1_hat = (1.0/(N/3))*np.dot((Xtr_w1-m2_hat).T,(Xtr_w1-m2_hat))

m3_hat = (1.0/(N/3))*np.sum(Xtr_w2, axis = 0)
S2_hat = (1.0/(N/3))*np.dot((Xtr_w2-m3_hat).T,(Xtr_w2-m3_hat))

S_hat = (1.0/3.0)*(S0_hat + S1_hat + S2_hat)
```

Part ii: Calculation of Mahalanobis distance on the test set using the estimated mean of each class

In [22]:

```python
inv_S = np.linalg.inv(S_hat)
dm_0 = np.sqrt(np.sum(np.dot((Xte-m1_hat), inv_S)*(Xte-m1_hat), axis = 1))
dm_1 = np.sqrt(np.sum(np.dot((Xte-m2_hat), inv_S)*(Xte-m2_hat), axis = 1))
dm_2 = np.sqrt(np.sum(np.dot((Xte-m3_hat), inv_S)*(Xte-m3_hat), axis = 1))

# Then, using the obtained euclidean distances, I classify the objects.
dm_matrix = np.stack((dm_0, dm_1, dm_2), axis = 1)
Mahal_result = np.argmin(dm_matrix, axis = 1)
```

Part iii: Use the Bayesian classifier to classify the points of X1 based on the ML estimates computed before

In [23]:

```python
def multivariate_normal_pdf_v2(x, mean, sigma):
    l = x.shape[1]
    det_S = np.linalg.det(sigma)
    norm_const = 1.0/((2.0*np.pi)**(l/2.0)*np.sqrt(det_S))
    inv_S = np.linalg.inv(sigma)
    a1    = np.sum(np.dot(x-mean, inv_S)*(x-mean), axis = 1)

    return norm_const*np.exp(-0.5*a1)
```

In [24]:

```python
baydis_x1 = Pw0*multivariate_normal_pdf_v2(Xte, m1_hat,S_hat)

baydis_x2 = Pw1*multivariate_normal_pdf_v2(Xte, m2_hat,S_hat)

baydis_x3 = Pw2*multivariate_normal_pdf_v2(Xte, m3_hat,S_hat)


de_matrix = np.stack((baydis_x1, baydis_x2, baydis_x3), axis = 1)
Bayes_result = np.argmax(de_matrix, axis = 1)

print(Bayes_result)
```

```
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0
 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 2 0 1 1 1 1 1
 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
 1 1 2 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2
 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2
 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2]
```

Part iv: To compute the error probability we compare the classification results with the reference matrix

In [25]:

```python
#Error Bayesian classifier probability
error_bayesian = 1-np.sum(Bayes_result == yte.flatten())/N # Bayes_result = yte.flatten()

#Error probability of Mahalanobis
error_mahalanobis = 1-np.sum(Mahal_result  == yte.flatten())/N

#print(error_bayesian)
print(error_bayesian)
#print(error_mahalanobis)
print(error_mahalanobis)

#The probablity error for bayesian and mahalanobis is performing equally
```

```
0.05500000000000005
0.05500000000000005
```

Comparing

*it is possible to observe that the error using both methods is the same due to the same probability that each class have