Assigment

Q1. $(x|\alpha, b) = \frac{b}{\Gamma(\alpha)} x^{\alpha-1} e^{-bx}$ $\qquad \alpha, b, x > 0$

The mean is calculated first

$$\int_0^{+\infty} \lambda \frac{1}{\Gamma(\alpha)} b^\alpha \lambda^{\alpha-1} \exp(-\lambda b) d\lambda = \frac{b^\alpha}{\Gamma(\alpha)} \int_0^{+\infty} \lambda^\alpha \exp(-\lambda b) d\lambda$$

$$= \frac{b^\alpha}{\Gamma(\alpha)} \int_0^\infty \left(\frac{v}{b}\right)^\alpha \exp(-v) \frac{1}{b} dv$$

$$= \frac{1}{\Gamma(\alpha) b} \int_0^\infty v^\alpha \exp(-v) dv$$

Knowing that $\Gamma(a+1) = a\Gamma(a)$ $\qquad = \frac{1}{\Gamma(\alpha) b} \Gamma(\alpha+1) = \frac{\alpha}{b}$

To calculate $E[\lambda^2]$:

$$\int_0^\infty \lambda^2 \frac{1}{\Gamma(\alpha)} b^\alpha \lambda^{\alpha-1} \exp(-\lambda b) d\lambda = \frac{b^\alpha}{\Gamma\alpha} \int_0^\infty \lambda^{\alpha+1} \exp(-b\lambda) d\lambda$$

$$= \frac{b^\alpha}{\Gamma(\alpha)} \int_0^\infty \left(\frac{v}{b}\right)^{\alpha+1} \exp(-v) \frac{1}{b} dv$$

$$= \frac{1}{\Gamma(\alpha) b^2} \int_0^\infty v^{\alpha+1} \exp(-v) dv$$

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

According to $Var[\lambda] = E[\lambda^2] - E[\lambda]^2 = \frac{\alpha(\alpha+1)}{b^2} - \left(\frac{\alpha}{b}\right)^2 = \frac{\alpha}{b^2}$

In order to find the mode of a gama distribution, the maximum of the PDF should be calculated, then it is neccessary to calculate the derivative respect to λ

$$\frac{d}{d\lambda} \left| \frac{1}{\Gamma(\alpha)} b^\alpha \lambda^{\alpha-1} \exp(-b\lambda) \right| = [(\alpha-1) - b\lambda] \frac{1}{\Gamma(\alpha)} b^\alpha \lambda^{\alpha-2} \exp(-\lambda b)$$

Its maximum is at $\lambda = \frac{\alpha-1}{b}$ so gamma distribution has mode at $Gam(\lambda|\alpha, b) = \frac{a-1}{b}$

Q_2 $\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{bmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1}+D^{-1}CMBD^{-1} \end{bmatrix}$ $M=(A-BD^{-1}C)^{-1}$

Knowing that $XX^{-1}=I$

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

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

$-AMBD^{-1}+BD^{-1}+D^{-1}CMBD^{-1}B=0$

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

$I=AM-CMBD^{-1}$

$I=M(A-CBD^{-1})$

$M=(A-D^{-1}BC)^{-1}$

$CM+D^{-1}CMD=0$

$-CMBD^{-1}D+DD^{-1}+DD^{-1}CMBD^{-1}=I$

Q3. $$R = \begin{pmatrix} \Lambda + A^T L A & -A^T L \\ -LA & L \end{pmatrix}$$

by the property in the previous question.

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

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

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

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

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

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

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

Then $\operatorname{cov}(z) = R^{-1} = \begin{pmatrix} \Lambda^{-1} & \Lambda^{-1} A^T \\ A\Lambda^{-1} & L^{-1} A \Lambda^{-1} A^T \end{pmatrix}$

Q4.

$P(x) = N(x \mid \mu_x, \Sigma_x)$ $\qquad y = x + z$

$P(z) = N(z \mid \mu_z, \Sigma_z)$

Knowing that $E[x] = x$

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

Then

$$\mu_{y|x} = E[x] + E[z] \qquad \Sigma_{y|x} = cov[x] + cov[z] = \Sigma_z$$
$$= x + \mu_z$$

$$P(y|x) = N(y \mid x + \mu_z, \Sigma_z)$$

by comparing

$$P(x) = N((x \mid \mu), \Lambda^{-1})$$
$$P(y|x) = N(y \mid Ax + b, L^{-1}$$
$$P(y) = N(y + A\mu + b, L^{-1} + A\Lambda^{-1}A^T)$$

$\mu = \mu_x \qquad b = \mu_z \qquad L^{-1} = \Sigma_z$

$A = I \qquad \Lambda^{-1} = \Sigma_x$

$$P(y) = N(y \mid \mu_x + \mu_z, \Sigma_z + A\Sigma_x A^T)$$
$$= N(y \mid \mu_x + \mu_z, \Sigma_z + \Sigma_x)$$

Q5

In this question it is necessary to write the joint distribution $p(x,y)$ and then combined it to obtain marginal distribution $p(y)$

The quadratic form of $p(x,y)$ in exponential form is:

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

terms with x are:

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

while integrating over x, it is possible to see that the first term dissappeared into a constant, then the remaining terms should be extracted

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

By using the inverse of $y^T y$, the covariance matrix can be obtained by:

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

By using woodbury inversion formula:

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

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

So

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

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

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

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

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

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

by using woodbury inversion method:

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

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

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

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

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

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

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

$$= A\mu + b$$

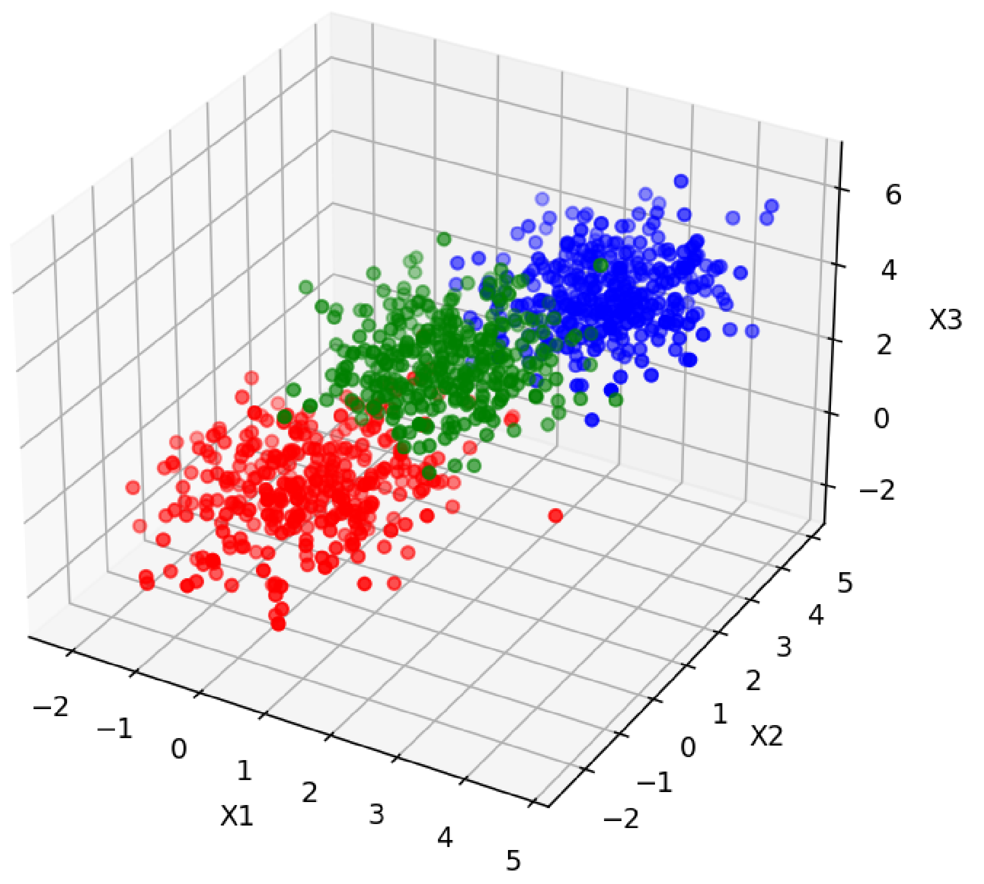
Question 2.6

In [1]:
```python
# Loading of relevant libraries
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

import seaborn as sns
```

In [2]:
```python
N = 1000

np.random.seed(4)

Pw0 = Pw1 = Pw2 = 1/3       # class a priori probabilities are equal

m0 = np.array([0, 0, 0])
m1 = np.array([1, 2, 2])
m2 = np.array([3, 3, 4])


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]])
```

In [3]:
```python
## training set

Xtr_w0 = np.random.multivariate_normal(m0, S1, 333)  # vectors for class_0
ytr_w0 = 0*np.ones((333, 1))                        # labels for class_0

Xtr_w1 = np.random.multivariate_normal(m1, S2, 333)  # vectors for class_1
ytr_w1 = 1*np.ones((333, 1))                        # labels for class_1

Xtr_w2 = np.random.multivariate_normal(m2, S3, 333)  # vectors for class_2
ytr_w2 = 2*np.ones((333, 1))                        # labels for class_2

# collection in a single set for data and labels

Xtr = np.concatenate((Xtr_w0, Xtr_w1, Xtr_w2), axis = 0)
ytr = np.concatenate((ytr_w0, ytr_w1, ytr_w2), axis = 0)



## test set

Xte_w0 = np.random.multivariate_normal(m0, S1, 333)    # vectors for class_0
yte_w0 = 0*np.ones((333, 1))                          # labels for class_0

Xte_w1 = np.random.multivariate_normal(m1, S2, 333)    # vectors for class_1
yte_w1 = 1*np.ones((333, 1))                          # labels for class_1

Xte_w2 = np.random.multivariate_normal(m2, S3, 333)    # vectors for class_2
yte_w2 = 2*np.ones((333, 1))                          # labels for class_2

# collection in a single set for data and labels

Xte = np.concatenate((Xte_w0, Xte_w1, Xte_w2), axis = 0)
yte = np.concatenate((yte_w0, yte_w1, yte_w2), axis = 0)
```

In [4]:
```python
# data ploting

%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 = "r", label = "Class 0")
ax.scatter(Xtr_w1[:,0], Xtr_w1[:,1], Xtr_w1[:,2], marker = "o", color = "g", label = "Class 1")
ax.scatter(Xtr_w2[:,0], Xtr_w2[:,1], Xtr_w2[:,2], marker = "o", color = "b", label = "Class 2")
ax.set_xlabel('X1')
ax.set_ylabel('X2')
ax.set_zlabel('X3')

plt.show()
```



In [5]:
```python
# question -1 : Calculation of ML estimates

m0_hat = (1.0/(N/3))*np.sum(Xtr_w0, axis = 0)
S0_hat = (1.0/(N/3))*np.dot((Xtr_w0-m0_hat).T,(Xtr_w0-m0_hat))

m1_hat = (1.0/(N/3))*np.sum(Xtr_w1, axis = 0)
S1_hat = (1.0/(N/3))*np.dot((Xtr_w1-m1_hat).T,(Xtr_w1-m1_hat))

m2_hat = (1.0/(N/3))*np.sum(Xtr_w2, axis = 0)
S2_hat = (1.0/(N/3))*np.dot((Xtr_w2-m2_hat).T,(Xtr_w2-m2_hat))

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

In [6]:
```python
# Question - 2
# Mahalanobis distance calculation on the test set from the estimate mean of each class

inv_S = np.linalg.inv(S_hat)
dm_0 = np.sqrt(np.sum(np.dot((Xte-m0_hat), inv_S)*(Xte-m0_hat), axis = 1))
dm_1 = np.sqrt(np.sum(np.dot((Xte-m1_hat), inv_S)*(Xte-m1_hat), axis = 1))
dm_2 = np.sqrt(np.sum(np.dot((Xte-m2_hat), inv_S)*(Xte-m2_hat), axis = 1))

# Classification based on the calculated euclidean distances

dm_matrix = np.stack((dm_0, dm_1, dm_2), axis = 1)
Mahal_result = np.argmin(dm_matrix, axis = 1)
```

In [7]:
```python
#Question - 3
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 [8]:
```python
baydis_x1 = Pw0*multivariate_normal_pdf_v2(Xte, m0_hat,S_hat)

baydis_x2 = Pw1*multivariate_normal_pdf_v2(Xte, m1_hat,S_hat)


baydis_x3 = Pw2*multivariate_normal_pdf_v2(Xte, m2_hat,S_hat)


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

In [9]:
```python
# Question - 4
# To compute the error probability the classification results are compare with the reference matrix


#error_bayesian clasifier

error_bayesian = 1-np.sum(Bayes_result == yte.flatten())/N

#error_mahalanobis

error_mahalanobis = 1-np.sum(Mahal_result  == yte.flatten())/N

#print(error_bayesian)

print(error_bayesian)

#print(error_euclidean)

print(error_mahalanobis)
```

```
0.05900000000000005
0.05900000000000005
```

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