ML ASSIGNMENT

Name - Ritik Gautam
Roll No - B180630CS

Question 2

Q. Consider the problem of classifying 10 samples from the Question 1 table of data. Assume that the underlying distributions are normal.

```
2.a Assume the prior probabilities of the first two categories are equal and is equal to 1/2 and that of the third category is zero.
    Design a dichotomizer for those two categories using the feature x1 alone.
2.b Determine the percentage of points misclassified.
2.c Repeat the above two steps, but now use the two features x1 and x2.
2.d Repeat again, with all the three features taken.
2.e Compare your results and conclude.
2.f Classify the points (1,2,1)t, (5,3,2)t, (0,0,0)t, (1,0,0)t using each feature vector mentioned above and compare the results.
```

Solution:

From Q1 we can use the data and the discriminant function as given below:

```
# Import Libraries that we need
import numpy as np
from numpy import log
from numpy.linalg import inv as inverse, det as determinant
```

```
# Given input
n = 3
data = [
        # ω1
        np.array([
            [-5.01, -8.12, -3.68],
            [-5.43, -3.48, -3.54],
            [1.08, -5.52, 1.66],
            [0.86, -3.78, -4.11],
            [-2.67, 0.63, 7.39],
            [4.94, 3.29, 2.08],
            [-2.51, 2.09, -2.59],
            [-2.25, -2.13, -6.94],
            [5.56, 2.86, -2.26],
            [1.03, -3.33, 4.33]
        ]),
        # ω2
        np.array([
            [-0.91, -0.18, -0.05],
            [1.30, -2.06, -3.53],
            [-7.75, -4.54, -0.95],
            [-5.47, 0.50, 3.92],
            [6.14, 5.72, -4.85],
            [3.60, 1.26, 4.36],
            [5.37, -4.63, -3.65],
            [7.18, 1.46, -6.66],
            [-7.39, 1.17, 6.30],
            [-7.50, -6.32, -0.31]

        ]),

        # ω3
        np.array([
            [5.35, 2.26, 8.13],
            [5.12, 3.22, -2.66],
            [-1.34, -5.31, -9.87],
            [4.48, 3.42, 5.19],
            [7.11, 2.39, 9.21],
```

```
            [7.17, 4.33, -0.98],
            [5.75, 3.97, 6.65],
            [0.77, 0.27, 2.41],
            [0.90, -0.43, -8.71],
            [3.52, -0.36, 6.43]
        ])
    ]


#find mean
mean = []
for i in range(len(data)):
  mean.append([sum(x)/len(x) for x in zip(*data[i])])
mean = np.array(mean)

#find covariance
covariance = []
for i in range(len(data)):
    covariance.append(np.cov(data[i].T))
covariance = np.array(covariance)

P = [1/2, 1/2, 0]


#The discriminant function is given below
def discriminant_function(i: int, x: np.array, P: list):
    if P[i] == 0:
        return -np.inf

    # finding dimension of input x
    dimention = x.shape[0]
    # Get the mean values based on given dimention
    mean_dimention = mean[:, 0:dimention]
    # Get the covariance values based on given dimention
    covariance_dimention = covariance[:, 0:dimention, 0:dimention]
    temp = np.matmul(inverse(covariance_dimention[i]), (x - mean_dimention[i]))
    #discriminant function
    res = -0.5 * np.matmul((x - mean_dimention[i]).T, temp) -0.5 * dimention * log(2 * np.pi) \
```

```
    - 0.5 * log(determinant(covariance_dimention[i])) + log(P[i])
    return res
```

This question states that we need to classify between 2 classes using priors(P) as [1/2, 1/2, 0]

```
# Set the priors to [1/2, 1/2, 0]
P = [1/2, 1/2, 0]
```

2.a and 2.b

Given that we need to design a dichotomizer based on feature x1 alone and find misclassification percentage. For that, we write a classification_function(f) that classify for each data point whether it is $\omega 1$, $\omega 2$ or $\omega 3$ using f features.

f = 1 -> using feature x1,

f = 2 -> using features x1 & x2,

f = 3 -> using features x1, x2 & x3

```
# Classify the data points based on f type of features
def classification_function(f: int):
    misclassification_rates = []
    for i in range(n):
        print('\n')
        print("CLASS ω%d" % (i + 1))
        print("."*12)
        misclassification_count = 0
        total_count = 0
        for x in data[i]:
            print(x, "is classified as ", end='\t')
            #finding discriminant values for all samples
            discriminant_value = np.array([discriminant_function(j, x[0:f], P) for j in range(n)])
            res = np.argmax(discriminant_value)
            print("ω%d" % (res + 1))
            if res != i:
                misclassification_count += 1
            total_count += 1
```

```
        misclassification_rate = (misclassification_count/total_count) * 100
        print("Misclassification rate: ", misclassification_rate, "%")
        misclassification_rates.append(misclassification_rate)

    misclassification_rates = np.round(np.mean(misclassification_rates), 3)

    print("Overall misclassification rate: ", misclassification_rates, "%")


# Classify for only x1 feature
classification_function(1)
```

```
CLASS ω1
............
[-5.01 -8.12 -3.68] is classified as    ω1
[-5.43 -3.48 -3.54] is classified as    ω2
[ 1.08 -5.52  1.66] is classified as    ω1
[ 0.86 -3.78 -4.11] is classified as    ω1
[-2.67  0.63  7.39] is classified as    ω1
[4.94 3.29 2.08] is classified as       ω2
[-2.51  2.09 -2.59] is classified as    ω1
[-2.25 -2.13 -6.94] is classified as    ω1
[ 5.56  2.86 -2.26] is classified as    ω2
[ 1.03 -3.33  4.33] is classified as    ω1
Misclassification rate:  30.0 %


CLASS ω2
............
[-0.91 -0.18 -0.05] is classified as    ω1
[ 1.3  -2.06 -3.53] is classified as    ω1
[-7.75 -4.54 -0.95] is classified as    ω2
[-5.47  0.5   3.92] is classified as    ω2
[ 6.14  5.72 -4.85] is classified as    ω2
[3.6  1.26 4.36] is classified as       ω1
[ 5.37 -4.63 -3.65] is classified as    ω2
[ 7.18  1.46 -6.66] is classified as    ω2
[-7.39  1.17  6.3 ] is classified as    ω2
[-7.5  -6.32 -0.31] is classified as    ω2
Misclassification rate:  30.0 %
```

```
CLASS ω3
............
[5.35 2.26 8.13] is classified as      ω2
[ 5.12  3.22 -2.66] is classified as    ω2
[-1.34 -5.31 -9.87] is classified as    ω1
[4.48 3.42 5.19] is classified as      ω2
[7.11 2.39 9.21] is classified as      ω2
[ 7.17  4.33 -0.98] is classified as    ω2
[5.75 3.97 6.65] is classified as      ω2
[0.77 0.27 2.41] is classified as      ω1
[ 0.9  -0.43 -8.71] is classified as    ω1
[ 3.52 -0.36  6.43] is classified as    ω1
Misclassification rate:  100.0 %
Overall misclassification rate:  53.333 %
```

2.c

We need to use x1 & x2 features, so call the function with 2 as argument.

```
# Classify for x1 & x2
classification_function(2)
```

```
CLASS ω1
............
[-5.01 -8.12 -3.68] is classified as    ω1
[-5.43 -3.48 -3.54] is classified as    ω2
[ 1.08 -5.52  1.66] is classified as    ω1
[ 0.86 -3.78 -4.11] is classified as    ω1
[-2.67  0.63  7.39] is classified as    ω2
[4.94 3.29 2.08] is classified as      ω2
[-2.51  2.09 -2.59] is classified as    ω2
[-2.25 -2.13 -6.94] is classified as    ω1
[ 5.56  2.86 -2.26] is classified as    ω2
[ 1.03 -3.33  4.33] is classified as    ω1
Misclassification rate:  50.0 %
```

```
CLASS ω2
............
[-0.91 -0.18 -0.05] is classified as    ω1
[ 1.3  -2.06 -3.53] is classified as    ω1
[-7.75 -4.54 -0.95] is classified as    ω2
[-5.47  0.5   3.92] is classified as    ω2
[ 6.14  5.72 -4.85] is classified as    ω2
[3.6  1.26 4.36] is classified as       ω1
[ 5.37 -4.63 -3.65] is classified as    ω2
[ 7.18  1.46 -6.66] is classified as    ω2
[-7.39  1.17  6.3 ] is classified as    ω2
[-7.5  -6.32 -0.31] is classified as    ω1
Misclassification rate:  40.0 %


CLASS ω3
............
[5.35 2.26 8.13] is classified as       ω2
[ 5.12  3.22 -2.66] is classified as    ω2
[-1.34 -5.31 -9.87] is classified as    ω1
[4.48 3.42 5.19] is classified as       ω1
[7.11 2.39 9.21] is classified as       ω2
[ 7.17  4.33 -0.98] is classified as    ω2
[5.75 3.97 6.65] is classified as       ω2
[0.77 0.27 2.41] is classified as       ω1
[ 0.9  -0.43 -8.71] is classified as    ω1
[ 3.52 -0.36  6.43] is classified as    ω1
Misclassification rate:  100.0 %
Overall misclassification rate:  63.333 %
```

2.d

We need to use all 3 features, so call the function with 3 as argument.

```
# Classify for x1, x2 & x3
classification_function(3)
```

```
CLASS ω1
............
[-5.01 -8.12 -3.68] is classified as    ω1
[-5.43 -3.48 -3.54] is classified as    ω1
[ 1.08 -5.52  1.66] is classified as    ω1
[ 0.86 -3.78 -4.11] is classified as    ω1
[-2.67  0.63  7.39] is classified as    ω2
[4.94 3.29 2.08] is classified as       ω1
[-2.51  2.09 -2.59] is classified as    ω1
[-2.25 -2.13 -6.94] is classified as    ω1
[ 5.56  2.86 -2.26] is classified as    ω2
[ 1.03 -3.33  4.33] is classified as    ω1
Misclassification rate:  20.0 %


CLASS ω2
............
[-0.91 -0.18 -0.05] is classified as    ω2
[ 1.3  -2.06 -3.53] is classified as    ω2
[-7.75 -4.54 -0.95] is classified as    ω2
[-5.47  0.5   3.92] is classified as    ω2
[ 6.14  5.72 -4.85] is classified as    ω2
[3.6  1.26 4.36] is classified as       ω1
[ 5.37 -4.63 -3.65] is classified as    ω2
[ 7.18  1.46 -6.66] is classified as    ω2
[-7.39  1.17  6.3 ] is classified as    ω2
[-7.5  -6.32 -0.31] is classified as    ω2
Misclassification rate:  10.0 %


CLASS ω3
............
[5.35 2.26 8.13] is classified as       ω1
[ 5.12  3.22 -2.66] is classified as    ω2
[-1.34 -5.31 -9.87] is classified as    ω1
[4.48 3.42 5.19] is classified as       ω1
[7.11 2.39 9.21] is classified as       ω1
[ 7.17  4.33 -0.98] is classified as    ω2
[5.75 3.97 6.65] is classified as       ω1
[0.77 0.27 2.41] is classified as       ω1
[ 0.9  -0.43 -8.71] is classified as    ω1
[ 3.52 -0.36  6.43] is classified as    ω1
```

```
Misclassification rate:  100.0 %
Overall misclassification rate:  43.333 %
```

2.e

We calculated the misclassification rate for all type of features as given below:

1. x1 - 53.33%
2. x1 & x2 - 63.33%
3. x1, x2 & x3 - 43.33%

Hence, order of better classifcation is:

3 features > 1 feature > 2 features.

So, if we take all 3 features(x1, x2 & x3) for classification then we get the best results.

```
#covariance matrices
covariance
```

```
array([[[ 14.38051111,   7.69537778,   4.12232222],
        [  7.69537778,  14.62312111,   3.90684   ],
        [  4.12232222,   3.90684   ,  19.72453778]],

       [[ 36.82933444,   9.98092667, -16.36675111],
        [  9.98092667,  13.16855111,   0.40905111],
        [-16.36675111,   0.40905111,  18.42121778]],

       [[  8.30475667,   7.44494667,  13.14957778],
        [  7.44494667,   8.56044889,  11.60861111],
        [ 13.14957778,  11.60861111,  47.28728889]]])
```

From above matrix we can observe that covariance with 3 features is highest, with 1 feature second highest while with 2 features its lowest. With 3 features, the features are more independent of each other and thus classifications are more accurate and the misclassification rate is the lowest.

2.f

New test data points are given. We have to classify them :

(1, 2, 1) , (5, 3, 2 , (0, 0, 0 , (1, 0, 0)

```
test_data_points = np.array([[1, 2, 1],
                             [5, 3, 2],
                             [0, 0, 0],
                             [1, 0, 0]
                             ])
```

```
# Classify test data based on f features
def classification_test(f):
    for x in test_data_points:
        print(x, "is classified as", end='\t')
        discriminant_value = np.array([discriminant_function(j, x[0:f], P) for j in range(n)])
        res = np.argmax(discriminant_value)
        print("ω%d" % (res + 1))
```

```
#First run classification test on feature x1 alone (i.e f = 1)
classification_test(1)
```

```
[1 2 1] is classified as        ω1
[5 3 2] is classified as        ω2
[0 0 0] is classified as        ω1
[1 0 0] is classified as        ω1
```

```
#Run classification test on features x1 & x2] (i.e f = 2)
classification_test(2)
```

```
[1 2 1] is classified as        ω1
[5 3 2] is classified as        ω2
```

```
[0 0 0] is classified as        ω1
```

```
#Run classification test on x1, x2 & x3 (i.e f = 3)
classification_test(3)
```

```
[1 2 1] is classified as        ω2
[5 3 2] is classified as        ω1
[0 0 0] is classified as        ω1
[1 0 0] is classified as        ω1
```

✓ 0s completed at 9:17 PM