

CS4023D Artificial Intelligence

Assignment 2

By Dev Sony, B180297CS

The question, report and source code can be found here.

[Github Repo](#)

Solution 1

Based on the formula given:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| + \ln P(\omega_i).$$

The function has been defined:

```
4 # Discriminant function as defined in the question
5 def discriminant_function(x, mean, cov, d, P):
6     # Checking if the dimensions turn out to be scalars in the case only 1 feature is being taken.
7     if d == 1:
8         output = -0.5*(x - mean) * (1/cov)
9         output = output * (x - mean)
10        output += -0.5*d*log(2*pi) - 0.5*log(cov)
11
12    else:
13        output = np.matmul(-0.5*(x - mean), np.linalg.inv(cov))
14        output = np.matmul(output, (x - mean).T)
15        output += -0.5*d*log(2*pi) - 0.5*log(np.linalg.det(cov))
16
17    # Adding Prior Probability
18    output += (log(P) if P != 0 else 0)
```

The variables can be configured based on the scenario. Here, it's assumed that prior probabilities are equally distributed and all features are taken:

```
66 # Arbitrary values
67 n = len(data)
68 P = [1/n for i in range(n)]
69 d = len(data[0][0])
70 g = [np.array([]) for _ in range(n)]
```

The input is the sample dataset, each set separated by the class they belong to as given below:

```

14 # Sample Data
15 data = [
16     # W1
17     np.array([
18         [-5.01, -8.12, -3.68],
19         [-5.43, -3.48, -3.54],
20         [1.08, -5.52, 1.66],
21         [0.86, -3.78, -4.11],
22         [-2.67, 0.63, 7.39],
23         [4.94, 3.29, 2.08],
24         [-2.51, 2.09, -2.59],
25         [-2.25, -2.13, -6.94],
26         [5.56, 2.86, -2.26],
27         [1.03, -3.33, 4.33]
28     ]),
29
30     # W2
31     np.array([
32         [-0.91, -0.18, -0.05],
33         [1.30, -2.06, -3.53],
34         [-7.75, -4.54, -0.95],
35         [-5.47, 0.50, 3.92],
36         [6.14, 5.72, -4.85],
37         [3.60, 1.26, 4.36],
38         [5.37, -4.63, -3.65],
39         [7.18, 1.46, -6.66],
40         [-7.39, 1.17, 6.30],
41         [-7.50, -6.32, -0.31]
42     ]),
43
44     # W3
45     np.array([
46         [5.35, 2.26, 8.13],
47         [5.12, 3.22, -2.66],
48         [-1.34, -5.31, -9.87],
49         [4.48, 3.42, 5.19],
50         [7.11, 2.39, 9.21],
51         [7.17, 4.33, -0.98],
52         [5.75, 3.97, 6.65],
53         [0.77, 0.27, 2.41],
54         [0.90, -0.43, -8.71],
55         [3.52, -0.36, 6.43]
56     ])
57 ]
58 ]

```

In order to classify the sample data, we first run the function through our sample dataset, classwise. On each sample, we find the class which gives the maximum output from its discriminant function.

A count and total count is maintained in order to find the success and failure rates.

```

82 # Taking each dataset from the classes in sample data
83 for j in range(n):
84     print("\nData classes should be classified as:", j+1)
85     total_count, count = 0, 0
86
87     # Taking x as dataset belonging to class j + 1
88     for x in data[j]:
89         g_values = [0 for g in range(n)] # Array for all discriminant function outputs.
90
91         # Itering through each class' discriminant function
92         for i in range(n):
93             g_values[i] = discriminant_function(x, means[i], cov[i], d, P[i])
94
95         # Now to output the maximum result
96         result = g_values.index(max(g_values)) + 1
97         print(x, "\twas classified as", result)
98         total_count, count = total_count + 1, (count + 1 if j == result - 1 else count)
99
100     print("Success Rate:", (count/total_count)*100,"%")
101     print("Fail Rate:", 100 - ((count/total_count)*100,"%")

```

Assuming that all classes have an equal prior probability (as per the configuration in the example picture), the following output is produced:

```

> Assg1 main* python Qn1/main.py

Data classes should be classified as: 1
[-5.01 -8.12 -3.68] was classified as 1
[-5.43 -3.48 -3.54] was classified as 1
[ 1.08 -5.52  1.66] was classified as 1
[ 0.86 -3.78 -4.11] was classified as 1
[-2.67  0.63  7.39] was classified as 2
[4.94  3.29  2.08] was classified as 3
[-2.51  2.09 -2.59] was classified as 1
[-2.25 -2.13 -6.94] was classified as 1
[ 5.56  2.86 -2.26] was classified as 3
[ 1.03 -3.33  4.33] was classified as 1
Success Rate: 70.0 %
Fail Rate: 30.0 %

Data classes should be classified as: 2
[-0.91 -0.18 -0.05] was classified as 2
[ 1.3 -2.06 -3.53] was classified as 3
[-7.75 -4.54 -0.95] was classified as 2
[-5.47  0.5  3.92] was classified as 2
[ 6.14  5.72 -4.85] was classified as 2
[3.6  1.26  4.36] was classified as 3
[ 5.37 -4.63 -3.65] was classified as 2
[ 7.18  1.46 -6.66] was classified as 2
[-7.39  1.17  6.3 ] was classified as 2
[-7.5 -6.32 -0.31] was classified as 2
Success Rate: 80.0 %
Fail Rate: 20.0 %

Data classes should be classified as: 3
[5.35 2.26 8.13] was classified as 3
[ 5.12 3.22 -2.66] was classified as 3
[-1.34 -5.31 -9.87] was classified as 3
[4.48 3.42 5.19] was classified as 3
[7.11 2.39 9.21] was classified as 3
[ 7.17 4.33 -0.98] was classified as 3
[5.75 3.97 6.65] was classified as 3
[0.77 0.27 2.41] was classified as 1
[ 0.9 -0.43 -8.71] was classified as 3
[ 3.52 -0.36 6.43] was classified as 3
Success Rate: 90.0 %
Fail Rate: 10.0 %

```

Solution 2

Part (a) and (b)

In order to match the question, the configuration variables are altered.

- (data-1) for n indicates that only 2 classes will be considered (the final class would not be considered as its Prior probability is 0, implying that it wouldn't appear.)
- The d value is changed to 1, indicating that only 1 feature will be used. (which is x_1)

```

78 # Configuration values
79 n = len(data) - 1
80 P = [0.5, 0.5, 0]
81 d = 1

```

The configuration parameters being passed are also changed.

- $x[0]$ indicates that only x^1 will be used.

- `means[i][0]` indicates that we need the mean only for x_1 .
- `cov[i][0][0]` indicates the variance of feature x_1 .

```

83 # Taking each dataset from the classes in sample data
84 for j in range(n):
85     print("\nData classes should be classified as:", j+1)
86     total_count, count = 0, 0
87
88     # Taking x as dataset belonging to class j + 1
89     for x in data[j]:
90         g_values = [0 for g in range(n)] # Array for all discriminant function outputs.
91
92         # Itering through each class' discriminant function
93         for i in range(n):
94             g_values[i] = discriminant_function(x[0], means[i][0], cov[i][0][0], d, P[i])
95
96         # Now to output the maximum result
97         result = g_values.index(max(g_values)) + 1
98         print(x, "\twas classified as", result)
99         total_count, count = total_count + 1, (count + 1 if j == result - 1 else count)
100
101 print("Success Rate:", (count/total_count)*100,"%")
102 print("Fail Rate:", 100 - ((count/total_count)*100,"%")

```

This results in the following output:

```

> Aassg1 main* python Qn2/a_b.py

Data classes should be classified as: 1
[-5.01 -8.12 -3.68] was classified as 1
[-5.43 -3.48 -3.54] was classified as 2
[ 1.08 -5.52  1.66] was classified as 1
[ 0.86 -3.78 -4.11] was classified as 1
[-2.67  0.63  7.39] was classified as 1
[4.94  3.29  2.08] was classified as 2
[-2.51  2.09 -2.59] was classified as 1
[-2.25 -2.13 -6.94] was classified as 1
[ 5.56  2.86 -2.26] was classified as 2
[ 1.03 -3.33  4.33] was classified as 1
Success Rate: 70.0 %
Fail Rate: 30.0 %

Data classes should be classified as: 2
[-0.91 -0.18 -0.05] was classified as 1
[ 1.3 -2.06 -3.53] was classified as 1
[-7.75 -4.54 -0.95] was classified as 2
[-5.47  0.5  3.92] was classified as 2
[ 6.14  5.72 -4.85] was classified as 2
[3.6  1.26  4.36] was classified as 1
[ 5.37 -4.63 -3.65] was classified as 2
[ 7.18  1.46 -6.66] was classified as 2
[-7.39  1.17  6.3 ] was classified as 2
[-7.5 -6.32 -0.31] was classified as 2
Success Rate: 70.0 %
Fail Rate: 30.0 %

```

Part (c)

Here, the configuration parameters are changed slightly.

- 'd's value is changed to 2, as now we are considering the first and second features.
- The matrix paramateres passed now include necessary values for the same reason.

```

78 # Configuration values
79 n = len(data) - 1
80 P = [0.5, 0.5, 0]
81 d = 2
82
83 # Taking each dataset from the classes in sample data
84 for j in range(n):
85     print("\nData classes should be classified as:", j+1)
86     total_count, count = 0, 0
87
88     # Taking x as dataset belonging to class j + 1
89     for x in data[j]:
90         g_values = [0 for g in range(n)] # Array for all discriminant function outputs.
91
92         # Itering through each class' discriminant function
93         for i in range(n):
94             g_values[i] = discriminant_function(x[0:2], means[i][0:2], cov[i][0:2, 0:2], d, P[i])

```

This results in the following output:

```

> Assg1 main* python Qn2/c.py

Data classes should be classified as: 1
[-5.01 -8.12 -3.68] was classified as 1
[-5.43 -3.48 -3.54] was classified as 2
[ 1.08 -5.52  1.66] was classified as 1
[ 0.86 -3.78 -4.11] was classified as 1
[-2.67  0.63  7.39] was classified as 2
[4.94 3.29 2.08] was classified as 2
[-2.51 2.09 -2.59] was classified as 2
[-2.25 -2.13 -6.94] was classified as 1
[ 5.56  2.86 -2.26] was classified as 2
[ 1.03 -3.33  4.33] was classified as 1
Success Rate: 50.0 %
Fail Rate: 50.0 %

Data classes should be classified as: 2
[-0.91 -0.18 -0.05] was classified as 1
[ 1.3 -2.06 -3.53] was classified as 1
[-7.75 -4.54 -0.95] was classified as 2
[-5.47  0.5  3.92] was classified as 2
[ 6.14  5.72 -4.85] was classified as 2
[3.6  1.26 4.36] was classified as 1
[ 5.37 -4.63 -3.65] was classified as 2
[ 7.18  1.46 -6.66] was classified as 2
[-7.39  1.17  6.3 ] was classified as 2
[-7.5 -6.32 -0.31] was classified as 1
Success Rate: 60.0 %
Fail Rate: 40.0 %

```

Part (d)

Here again, the configurations are changed in a similiar fashion as in (c).

- 'd' values is changed to 3 as all three features are now considered.
- The matrix paramaeteres are now passed without slicing as all values are important.

```

78     # Configuration values
79     n = len(data) - 1
80     P = [0.5, 0.5, 0]
81     d = 3
82
83     # Taking each dataset from the classes in sample data
84     for j in range(n):
85         print("\nData classes should be classified as:", j+1)
86         total_count, count = 0, 0
87
88         # Taking x as dataset belonging to class j + 1
89         for x in data[j]:
90             g_values = [0 for g in range(n)]          # Array for all discriminant function outputs.
91
92             # Itering through each class' discriminant function
93             for i in range(n):
94                 g_values[i] = discriminant_function(x, means[i], cov[i], d, P[i])

```

The results in the following output:

```

> Assg1 main* python Qn2/d.py

Data classes should be classified as: 1
[-5.01 -8.12 -3.68] was classified as 1
[-5.43 -3.48 -3.54] was classified as 1
[ 1.08 -5.52  1.66] was classified as 1
[ 0.86 -3.78 -4.11] was classified as 1
[-2.67  0.63  7.39] was classified as 2
[4.94  3.29  2.08] was classified as 1
[-2.51  2.09 -2.59] was classified as 1
[-2.25 -2.13 -6.94] was classified as 1
[ 5.56  2.86 -2.26] was classified as 2
[ 1.03 -3.33  4.33] was classified as 1
Success Rate: 80.0 %
Fail Rate: 20.0 %

Data classes should be classified as: 2
[-0.91 -0.18 -0.05] was classified as 2
[ 1.3  -2.06 -3.53] was classified as 2
[-7.75 -4.54 -0.95] was classified as 2
[-5.47  0.5  3.92] was classified as 2
[ 6.14  5.72 -4.85] was classified as 2
[3.6  1.26  4.36] was classified as 1
[ 5.37 -4.63 -3.65] was classified as 2
[ 7.18  1.46 -6.66] was classified as 2
[-7.39  1.17  6.3 ] was classified as 2
[-7.5  -6.32 -0.31] was classified as 2
Success Rate: 90.0 %
Fail Rate: 10.0 %

```