

# 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     output = np.matmul(-0.5*(x - mean), np.linalg.inv(cov))
7     output = np.matmul(output, (x - mean).T)
8     output += -0.5*d*log(2*pi) - 0.5*log(np.linalg.det(cov))
9     output += (log(P) if P != 0 else 0)
10
11     return output
```

The variables can be configured based on the scenario:

```
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 rates.

```

72 # Taking each dataset from the classes in sample data
73 for j in range(n):
74     print("\nData classes should be classified as:", j+1)
75     total_count, count = 0, 0
76     # Taking x as dataset belonging to class j+1
77     for x in data[j]:
78         g_values = [0 for g in range(n)] # Array for all discriminant function outputs.
79         # Itering through each class' discriminant function
80         for i in range(n):
81             g_values[i] = discriminant_function(x, means[i], cov[i], d, P[i])
82
83         # Now to output the maximum result
84         result = g_values.index(max(g_values)) + 1
85         print(x, "\twas classified as", result)
86         total_count, count = total_count + 1, (count + 1 if j == result - 1 else count)
87
88     print("Success Rate:", (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 %
```

```
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 %
```

```
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 %
```