CISC3024 Assignment 1

Experiments

In [12]:
```
import torch
import numpy as np
```

Calculate the two error rates respectively for the two classes ω_1 and ω_2, given by the means and a common covariance matrix. The decision boundary is pre-computed correctly and given as a parameter.

In [25]:
```
def calc_error_rate(data_num, means, cov, dec_bound):
    mean_1, mean_2 = means

    X_test_1 = np.random.multivariate_normal(mean_1, cov, data_num)
    X_test_2 = np.random.multivariate_normal(mean_2, cov, data_num)

    X1_X1 = 0
    X1_X2 = 0
    X2_X1 = 0
    X2_X2 = 0

    for xt_1 in X_test_1:
        x1, x2 = xt_1
        if x1 + x2 > dec_bound:
            X1_X2 += 1
        elif x1 + x2 < dec_bound:
            X1_X1 += 1

    for xt_2 in X_test_2:
        x1, x2 = xt_2
        if x1 + x2 > dec_bound:
            X2_X2 += 1
        elif x1 + x2 < dec_bound:
            X2_X1 += 1

    error_rate_1 = X1_X2 / (X1_X1 + X1_X2)
    error_rate_2 = X2_X1 / (X2_X1 + X2_X2)

    return error_rate_1, error_rate_2
```

Perform numerous calculations and seek an average among them to ensure convergency.

In [26]:
```
def calc_error_rate_accurate(data_num, means, cov, num_iter, dec_bound):
    error_rates_1 = []
    error_rates_2 = []

    for i in range(0, num_iter):
        error_rate_1, error_rate_2 = calc_error_rate(data_num, [mean_1, mean_2], cov, dec_bound)
        error_rates_1.append(error_rate_1)
        error_rates_2.append(error_rate_2)

    error_rate_1 = np.mean(error_rates_1)
    error_rate_2 = np.mean(error_rates_2)

    print(f"Error rate for class 1:{error_rate_1:.5f}\nError rate for class 2:{error_rate_2:.5f}")
```

Perform computation for two cases:

- Case 1:$\mu_2 = \begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix}$
- Case 1:$\mu_2 = \begin{bmatrix} 3.0 \\ 3.0 \end{bmatrix}$

In [27]:
```
data_num = 100
mean_1 = [1, 1]
mean_2 = [1.5, 1.5]
cov = [[1, 0], [0, 1]]

calc_error_rate_accurate(data_num, [mean_1, mean_2], cov, 100000, 2.5)
```

```
Error rate for class 1:0.36200
Error rate for class 2:0.36220
```

In [28]:
```
data_num = 100
mean_1 = [1, 1]
```

```
mean_2 = [3, 3]
cov = [[1, 0], [0, 1]]

calc_error_rate_accurate(data_num, [mean_1, mean_2], cov, 100000, 4)
```

```
Error rate for class 1:0.07864
Error rate for class 2:0.07866
```