Consider a two-class classification problem where the data in each class is modeled by a 2D Gaussian density \( N(\mu_1, \Sigma_1) \) and \( N(\mu_2, \Sigma_2) \).

Data Generation: using the parameters shown below, generate 60,000 random samples from \( N(\mu_1, \Sigma_1) \) and 140,000 samples from \( N(\mu_2, \Sigma_2) \) (i.e., 200,000 samples total). We will be referring to this data set as “data set A”.

\[
\begin{align*}
\mu_1 &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
\Sigma_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
\mu_2 &= \begin{bmatrix} 4 \\ 4 \end{bmatrix} \\
\Sigma_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\end{align*}
\]

Notation:

\[
\begin{bmatrix}
\mu_x \\
\mu_y
\end{bmatrix}
\begin{bmatrix}
\sigma_x^2 & 0 \\
0 & \sigma_y^2
\end{bmatrix}
\]

Note: you will be using the Box-Muller transformation to generate the samples from each distribution; please review “Generating Gaussian Random Numbers” for more information (posted on the course’s webpage). A link to the C code is provided on the webpage. Since the code generates samples from a 1D Gaussian distribution, you would need to call the Box-Muller function twice to generate 2D samples \((x, y)\); use \((\mu_x, \sigma_x)\) to generate the \(x\) sample and \((\mu_y, \sigma_y)\) to generate the \(y\) sample.

Note: \texttt{ranf()} is not defined in the standard library, you could use the simple implementation:

```c
/* ranf - return a random double in the [0,m] range.*/

double ranf(double m) {
    return (m*rand()/(double)RAND_MAX);
}
```

1. This experiment involves the samples from set A
   a. Design a Bayes classifier for minimum error to classify the samples from set A. Which discriminant (i.e., case I, II, or III) would you use in this experiment and why? How would you set the prior probabilities \( P(\omega_1) \) and \( P(\omega_2) \)?
   b. Plot both the Bayes decision boundary and generated samples on the same plot to better visualize how the Bayes rule would classify the data.
   c. Classify all 200,000 samples and report (i) the misclassification rate for each class separately (i.e., percentage of misclassified samples for each class) and (ii) the total misclassification rate (i.e., percentage of misclassified samples overall).
   d. Calculate the theoretical probability error (e.g., Bhattacharyya bound) and compare it with the misclassification rate from part (c).
Data Generation: using the parameters shown below, generate 40,000 random samples from $\mathcal{N}(\mu_1, \Sigma_1)$ and 160,000 samples from $\mathcal{N}(\mu_2, \Sigma_2)$ (i.e., 200,000 samples total). We will be referring to this data set as “data set B”.

\[
\begin{align*}
\mu_1 &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} & \Sigma_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \mu_2 &= \begin{bmatrix} 4 \\ 4 \end{bmatrix} & \Sigma_2 &= \begin{bmatrix} 4 & 0 \\ 0 & 8 \end{bmatrix}
\end{align*}
\]

2. Repeat part (1) experiments using the samples from set $\text{B}$. How do your results from this part compare with your results from part (1) and why?

3. Quite often, the Euclidean distance classifier (shown below but also discussed in the lecture) is used for various classification tasks without fully understanding that it is optimum only under certain assumptions. In this sense, the Euclidean distance classifier might not always be an optimum classifier. Classify the samples from set $\text{A}$ using the Euclidean distance classifier and compare your results with those obtained from part (1). Is the Euclidean distance classifier an optimum classifier for data set $\text{A}$? Explain.

\[
\mathcal{g}_i(\mathbf{x}) = -||\mathbf{x} - \mu_i||^2
\]

4. Repeat part (3) experiments using the samples from set $\text{B}$. Compare and discuss your results with those obtained from part (2). Is the Euclidean distance classifier an optimum classifier for data set $\text{B}$? Explain. How does its performance compare with that for set $\text{A}$?