CS 479/679 Pattern Recognition
Spring 2020 – Prof. Bebis
Programming Assignment 1 - Due: 2/25/2020

Consider a two-class classification problem where each class is modeled by a 2D Gaussian distribution $G(\mu_1, \Sigma_1)$ and $G(\mu_2, \Sigma_2)$.

1. Generate 100,000 samples from each distribution (i.e., 200,000 samples total) using the following parameters (i.e., each sample (x, y) can be thought as a feature vector):

$$\mu_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad\qquad \mu_2 = \begin{bmatrix} 4 \\ 4 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Notation:

$$\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

Note: this is not the same as "sampling" the 2D Gaussian functions; see "Generating Gaussian Random Numbers" which is posted on the course's webpage for more information on how to generate the samples using the **Box-Muller** transformation. A link to C code is also provided on the course webpage. Since the code generates samples for 1D distributions, you would need to call the function twice to get a 2D sample (x, y); use (μ_x, σ_x) for the x sample and (μ_y, σ_y) for the y sample.

Note: *ranf()* is not defined in the standard library and that you would need to implement it yourself using *rand()*; for example:

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

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

(m=1 in our case)

a. Assuming $P(\omega_1) = P(\omega_2)$
   i. Design a Bayes classifier for minimum error.
   ii. Plot the Bayes decision boundary **together** with the generated samples to better visualize and interpret the classification results.
   iii. Report (i) the number of misclassified samples for each class separately and (ii) the total number of misclassified samples.
   iv. **Grad Students Only**: Plot the Chernoff bound as a function of β and find the optimum β for the minimum.
   v. Calculate the Bhattacharyya bound.

b. Repeat part (a) for $P(\omega_1) = 0.2$ and $P(\omega_2) = 0.8$. For comparison purposes, use **exactly the same** 200,000 samples from (a).

2. Repeat (1.a) and (1.b) using the parameters below; you would of course need to generate new sample sets!

$$\mu_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mu_2 = \begin{bmatrix} 4 \\ 4 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 4 & 0 \\ 0 & 8 \end{bmatrix}$$

3. Repeat (2.b) (i.e., P(ω1) ≠ P(ω2)) using the **minimum-distance classifier** (which assumes equal priors) and compare your results (i.e., misclassified samples) with those obtained in part (2.b). For comparison purposes, use exactly the same 200,000 samples as in part 2.