CA 1: Classification with MLP updated by Backprop from Scratch and RBF Networks with Random Selected Center Neurons.

This assignment aims to explore classification on a set of data. The classification will first be achieved through MLP (Multi-Layer Perceptron) which would be updated by the Backpropagation following gradient descent. Then classification will be further achieved by the RBF network with randomly selected RBF centers.

Before you start the assignment, please read through the following guidelines so that you understand the requirements. Follow the guidelines strictly to avoid unnecessary mark deductions:

1. Implement your codes with Python (recommended) or MATLAB (or any other languages you deem comfortable with).
2. For questions that require you to show your code, make sure your codes are **clean** and **easily readable** (add meaningful comments, comments are **markable**). Embed your code into your answer with screenshots.
3. Your submission should be one single PDF file with the necessary code and results collated in a single file. Name your PDF file strictly to **"MATRICULATION NUMBER_YOUR NAME_CA1.pdf"**. Incorrect submission format or naming will result in a **direct** 20 points (out of 100) deduction.
4. Do submit your project on Canvas before the deadline: **5:59 pm (SGT), 7 March 2025**. There will be **3** submissions attempts restricted for every student.
5. Policy on late submission: the deadline is a strict one, so please prepare and plan early and carefully. Any late submission will be deducted 10 points (out of 100) for every 24 hours.
6. This is an **individual** project, do **NOT** share your solutions with others, we have zero tolerance for **plagiarism**.
7. You are **NOT** allowed to use the direct functions for each individual task/step (e.g., sklearn.neural_network.MLPClassifier for constructing MLPs or scipy.linalg.lstsq for finding the LS estimation result), but other than these, you may use the basic operations in any packages.

Part 1: Classification with MLP updated by Backpropagation following gradient descent (55%).

a. Before we begin the classification, let us generate the binary class data first. The data can be generated and stored with the “Generate_data.py” file. It will create one scatter plot of the data, and “.npy” and “.mat” files that stores the data. You may also generate with your own set of codes. Show the scatter plot of the generated data. Further, if you implement your own data generation code, show your code. Note that the ground-truth output of class 1 is 0, and the ground-truth output of class 2 is 1. (5%)

b. Next, implement a Multi-Layer Perceptron (MLP) for classification. The MLP is a 2-layer MLP with only ONE hidden layer. Its structure should be 2 input neurons, 3 hidden layer neurons, and 1 output neuron (binary output). You may select the activation function for all the neurons (e.g., ReLU activation). Show detailed implementation of the MLP. There are a total of 9 connections between the neurons. Initialize their weights randomly (e.g., follow a normal distribution), then plot the decision boundary of the initial MLP. To plot the decision boundary, you may wish to refer to the following: Webpage1, Webpage2, and Webpage3. With this initial MLP, what is the classification accuracy, precision, and recall of the classifier? What can you **observe** and **imply** from the results obtained? Plot the ROC curve for the classifier. (25%)

c. Subsequently, we update the weights of the MLP connections (9 of them) with the Backpropagation algorithm, following the spirit of gradient descent. Implement the update from scratch and update the weights over ALL data entries. You may select the cost/loss function you deemed suitable (e.g., squared error). Show your code in detail. Then plot the new decision boundary as in Part 1b. What is the classification accuracy, precision, and recall with the updated MLP? **Compare** the two decision boundaries and **discuss** the differences and the effects of your implemented update. Plot the ROC curve for the updated classifier and compare it with the ROC curve obtained from 1b. (25%)

Part 2: Classification with RBF Network with Random RBF Centers (45%).

a. With the same set of data as in Part 1, now we leverage the RBF network for this part of classification task. Assume that the Gaussian function is used for the hidden layer neurons with the width set as $\sigma = 1$. Further, we set that there are 3 hidden neurons, and that the 3 RBF centers are randomly set. The randomly set RBF centers should be situated within the range of all the data. The RBF neural network also has only 1 single output neuron. Show your code that constructs this RBF network and show the plot that highlights the randomly chosen RBF center. Subsequently, code and determine the weights from the hidden layer to the output layer, using the LS estimation algorithm implemented from scratch. Next, plot the decision boundary of this RBF network and determine its classification accuracy, precision, and recall. Compare and discuss the performance of this RBF network compared to the updated MLP after Part 1c. (30%)

b. Repeat Part 2a, now with 6 RBF centers that are randomly set. The hidden neurons still leverage the Gaussian function with the same width of $\sigma = 1$. Determine the new weights between the hidden layer and the output neuron. Plot the decision boundary for this RBF network and determine its classification accuracy, precision, and recall. Compare with the RBF network in Part 2a and discuss the effects of the numbers of hidden neuron on the performance of the RBF network. (15%)