**Cairo University Faculty of Computers and Artificial Intelligence**

**Third year (2020-2021) Supervised learning (Spring 2021)**

**Assignment 2**

**Programming Language:** Python

**Input:** Draw some points on a 2D grid for two classes, at least 6 points for each class, most of those points should be distinguishable as class 1 or class 2 (choose the points yourself). The coordinates (x, y) will be your input (x1, x2), the class that the point belongs to will be either 0 or 1. The output class is what your neural network should learn to classify. (NOTE: the drawing part is for yourself, you can do it on a paper or paint to determine where your points will be and to make sure that most of them can be distinguished)

**Requirement:** Build your own neural network using only python’s available built-in functions, without any libraries/imports (3 helper libraries written in the restriction part). The neural network should contain at least 2 layers: 1 hidden, 1 output.

An example: input 🡪**hidden layer** **1**🡪 hidden layer 1 output 🡪 **output layer** 🡪 output.

The number of neurons in a layer should be of a minimum 4. Do note that the number of neurons is **NOT** fixed, do not write your code based on a fixed number, it should be dynamic and changeable.

**Features:** Your input is as explained above, and the output is 0 or 1 depending on the class. Note that you threshold the output to classify, but you don’t do that in back propagation as you need the error value.

**Prediction:** The neural network will do the predictions for you. You just have to threshold the value after using the activation function to get the prediction. Compare the prediction with the original and get %.

**Summarized Steps:** You will use the equations that you have taken in the lecture to pass the input through your neural network layers. You will initialize the weights of the layers with random values. You have the equations to calculate the output for all the forward passes which is simple multiplication of the values and the input values/values from previous layer. Do note to save the output values as you will need them for the back propagation.

Your activation function for the final output layer will be the sigmoid function, and your error function will be MSE. For the back propagation you can also refer to the lecture to know how to implement it. You will propagate back the error and change the weights accordingly. Your learning rate should be set to 0.005, run the learning for 100 epoch. 1 epoch means a forward and backward propagation of all your samples.

**Restrictions:** No imports should be used except for **math**, **time** and **random** library are the exception and those are included for the use of ceil, floor, round, time and random. You should build the neural network and calculate everything that happens in it yourself using lists, arrays, for loops, etc… no shortcuts allowed.