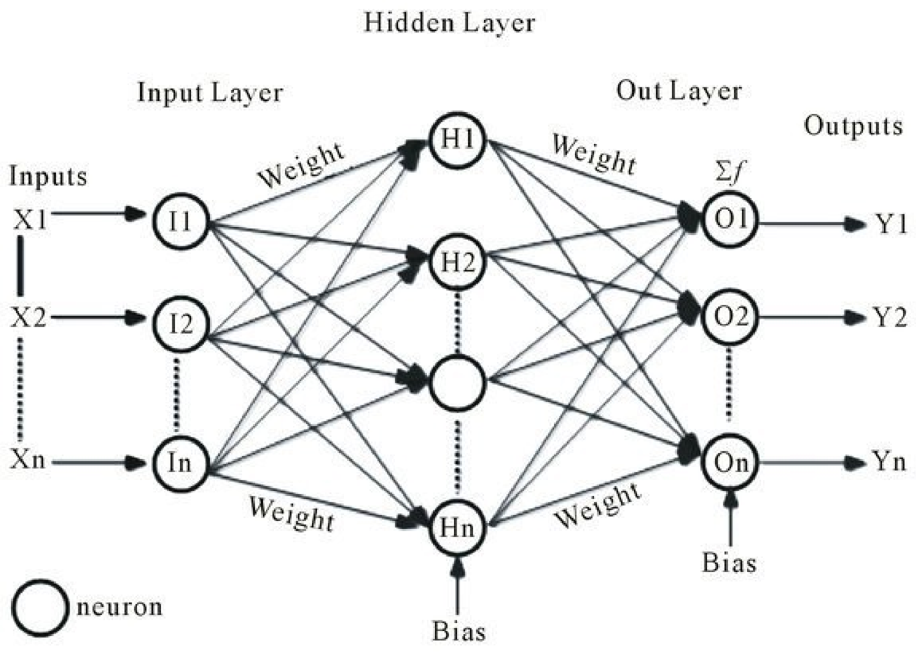
COP 701: Toy Neural Network

Instructions:

1. Upload your solution in a zip folder following the mentioned naming scheme i.e. <EntryNumber_Name_assignment-4>.zip.
2. In this assignment, you can use ~~either~~ C ~~or C++~~.
3. If you are copying the code from Github or peers, the consequences can range from getting an F grade in the course to the Disciplinary Committee.

Introduction:

A neural network can be defined as a computing system consisting of several neurons or computing elements that are highly interconnected and processes the input provided externally. This computing system was motivated by a biological neural network in the human brain which processes information. Fig-1 represents the architecture of the multi-layer perceptron based neural network, and the following are the components of a neural network.



MLP

1. Input Node (Input Layer): A block of nodes is called a layer. The input layer provides the input data points to the neural network and passes down the data to the hidden layers.
2. Hidden Node (Hidden Layer): In the hidden layer, the computation is done, the result is transferred from the previous layer to the following layer.
3. Activation Function: This function is imposed on the nodes to define the output with respect to input or set of input. Commonly used activation functions are Sigmoid, Tanh, Arctan, Leaky Relu and ReLu.
4. Output Node: This is the node that provides us with the output of the test data point.
5. Connection and Weights: In a neural network, every node in an i^{th} layer is connected to the jth layer, this is called a connection, and each such connection is associated with weight w_{ij}.

Hyperparameters:

1. Learning Rate: During the neural network training, the learning rate defines the size of the corrective steps that the model takes to adjust the error in each observation.
2. Cost Function: In this assignment, you would use either cross-entropy or mean squared error as a cost function.
3. Backpropagation: The backpropagation method alters the connection weights to minimise the error found during training. Typically, it calculates the derivative of the cost function.

Backpropagation Algorithm:

1. Batch Gradient Descent:
   In the batch gradient descent algorithm, we calculate the error for every data point belonging to the dataset, but only after training of the dataset is completed.
2. Stochastic Gradient Descent:
   In comparison to batch gradient descent, stochastic gradient descent updates the parameter for each training data point.
3. Mini-Batch Gradient Descent:
   Mini-Batch Gradient Descent algorithm splits the dataset into batches and performs an update after training of each of the batches.

In this assignment, you are expected to submit the following:

1. Implement a multi-layer perceptron (MLP) based neural network using C.
2. The implementation should be a parameter-based library. The client or user can define the activation function, cost function and backpropagation technique using some arguments and then build the <library-file>.so file. The activation functions can either be Sigmoid or Tanh, or ReLu. The backpropagation algorithm can be either Batch gradient descent or stochastic gradient descent or mini-batch gradient descent. The cost function can be either cross-entropy or mean squared error
3. Implement an MLP classifier using Scikit learn library in python.
4. Design multiple multilayer perceptrons with different activation functions. Plot a loss function graph, wherein the x-axis would be your number of iterations, and the y-axis would be the loss function value with every iteration.
5. The loss function graph should have multiple curves corresponding to the Scikit-learn version of MLP, your implementation of the MLP classifier with different activation functions such as Sigmoid or Tanh, or ReLu. You can have two versions of the loss function graph, one with cross-entropy as a cost function and one with mean squared error as a cost function.
6. Record the accuracy of the classifier and also state the observation in the report.
7. Simulate gradient vanishing problem using a suitable cost function, activation function and backpropagation algorithm. Can we get rid of gradient vanishing problems? If yes, how and If no, why?

References:

1. https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc
2. https://builtin.com/data-science/gradient-descent
3. https://medium.com/@devalshah1619/activation-functions-in-neural-networks-58115cda9c96
4. https://en.wikipedia.org/wiki/Artificial_neural_network
5. http://deeplearning.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/
6. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html