EE 550
Artificial Neural Network
HW Project #3

Implementation of Multilayer Perceptron

Due on: April 9th, 2018

This project requires the simulation of the multilayer perceptron model (MLP) we have discussed in class. You can use any of the programming languages that you are familiar with.
You can use graphical tools for the required outputs of the models.

1) First simulate the MLP model that can be used with arbitrary number of layers and neurons in each layer.

   Generate a data set for the binary XOR function. You will have a two input one output model. Your training pattern set will have 4 patterns.

   Choose a 3 layer network with arbitrary number of nodes in hidden layer (8-9 nodes would be sufficient for hidden layers..)

   Plot the total error (cost) function vs number of epochs (an epoch is defined as going over the training set once).

   Once convergence is achieved, test the network with each sample pattern. Show the outputs for each input pattern.

2) Function approximation: Pick a nonlinear function like f(x) = Sin (x) where is given in radians.
   Generate a data set for x between 0 and 2*Pi. You should arbitrarily generate at least 40 data points.

   Implement 3, and 4 layer networks. For each case you can arbitrarily choose the number of neurons in each layer (10-12 neurons should be sufficient)

   Plot the total cost function versus number of epochs.

Once the cost function reaches a low threshold value, stop the learning and
Test your network arbitrary test inputs between 0 and 2*Pi and plot the approximate graph on top the desired function Sin(x).

3) From the data repository site archive.ics.uci.edu download the Iris data set. This data set has 150 samples and 4 attributes. Choose an MLP model with 3 and 4 layers. Choose arbitrarily 125 samples from this data set as your training samples. Train the ANN with these samples. Plot the total error versus number of epochs. Once convergence is achieved (total error below a certain threshold) stop the learning algorithm, and test your network with the remaining 25 patters. Plot the input and output for each case, and determine if the network classifies the patterns correctly.

You can use a sigmoid function as an activation function of the neurons you use in the MLP models for all the above cases.
You can use a threshold value at the output of each neuron in the above models.
You can implement the momentum term in the learning algorithm to get better convergence.