Howard Goldowsky
November 27, 2018
Problem Set #4, Artificial Neural Network to Classify Fisher's Iris Plant Database

Discussion

Figure 1 shows a diagram of my ANN design. (While I show only two nodes in the hidden layer here, I needed six nodes in the hidden layer to approach graceful training curves.) Table 1 lists the variables used in my back-propagation algorithm. The variable names in Table 1 were adopted from the notation used in Chapter Two of Michael Nielsen's text, *Neural Networks and Deep Learning* (http://neuralnetworksanddeeplearning.com/chap2.html).

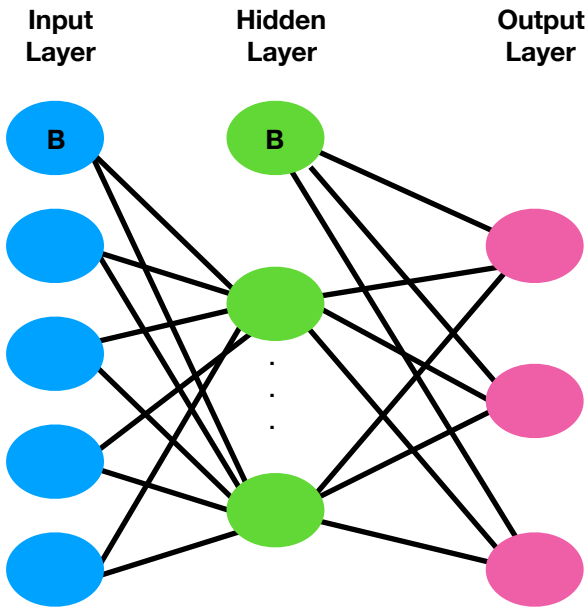

Figure 1

Variable Name	Variable Dimensions	Explanation
a1	[samples] X [features + 1]	input layer, with bias
Theta1	[hidden units] X [features + 1]	coefficients between input and hidden layers
Theta2	[classifiers] X [hidden units + 1]	coefficients between hidden layer and output layers
z2	[hidden units] X [samples]	hidden layer nodes before Sigmoid
a2	[hidden units] X [samples]	hidden layer nodes after Sigmoid
a22	[samples] X [hidden units + 1]	hidden layer nodes after Sigmoid, with bias
z3	[classifiers] X [samples]	output layer before Sigmoid
a3	[classifiers] X [samples]	output layer after Sigmoid

Table 1

As discussed in class, I used 60% training data, 20% validation data, and 20% test data.

To test the validation data and to determine how many iterations to run, I wrote a script called ValidationTestScript.m. This validation script sweeps the number of iterations passed to the main ANN.m, from 100 to 5000, in increments of 100. It then plots the accuracy data as a function of number of iterations, so the user can determine the proper number of iterations to use for training.

I found that using six nodes in the hidden layer, a learning rate of 0.33, and about 800 iterations produced good 97% to 100% accuracy rates on the validation data. The user can tune the number of iterations, the number of hidden nodes (line 33 of ANN.m), the learning rate (line 11 of the validation test script), or any combination of these parameters.

If I were to run this in production, I would halt the validation script when the validation accuracy begins to dip, and then freeze the weights at that time.

Run ValidationTestScript.m to reproduce my validation procedure. An example of this script's output for when the learning rate is 0.33 is given in Figure 2.

The validation script will call
[validation_error, test_error, training_error] = ANN(learnRate, numIterations), which is the main implementation of the neural network.

ANN.m is a short main routine that calls subroutines in a step-by-step manner to perform the following steps:

1. Pre-process data, format data, break into feature data set and truth labels, normalize all features to between [0,1].
2. Visualize data. This is a simple routine to view three of the four features for each data point and to get a feel for the data. See Figure 3 for a sample plot of unnormalized data.
3. Break data into training (60%), validation (20%), and test (20%) sets.
4. Initialize the ANN architecture and initial random weights.
5. Train the weights.
6. Run the neural net in the forward direction and report test results.

Instructions

Please make sure your Matlab path is set correctly to include all files downloaded and unzipped.

To run the validation script that calls ANN.m, type "ValidationTestScript" at the Matlab command line.

To run the ANN as a stand-alone function, type the following into the Matlab command line: "[validation_error, test_error, training_error] = ANN(learnRate, numIterations)". Be sure to define values for learnRate and numIterations before running ANN.m stand-alone.

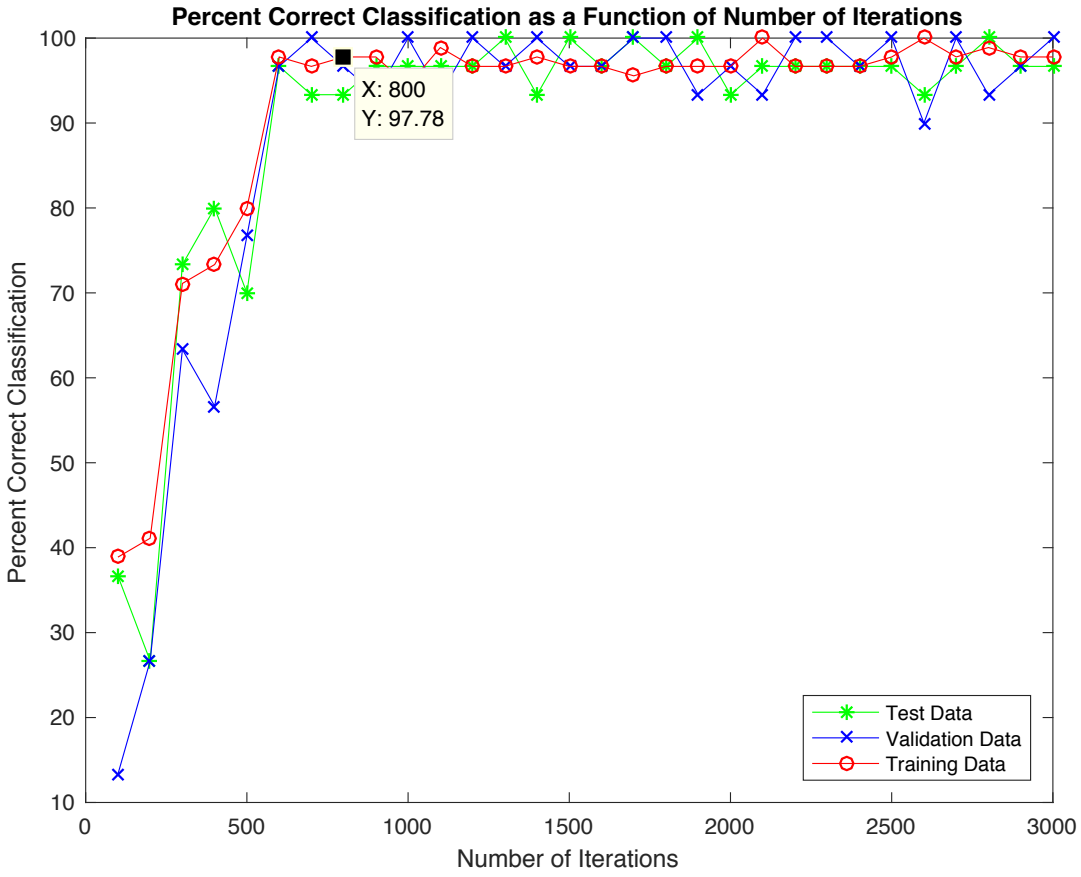

Figure 2

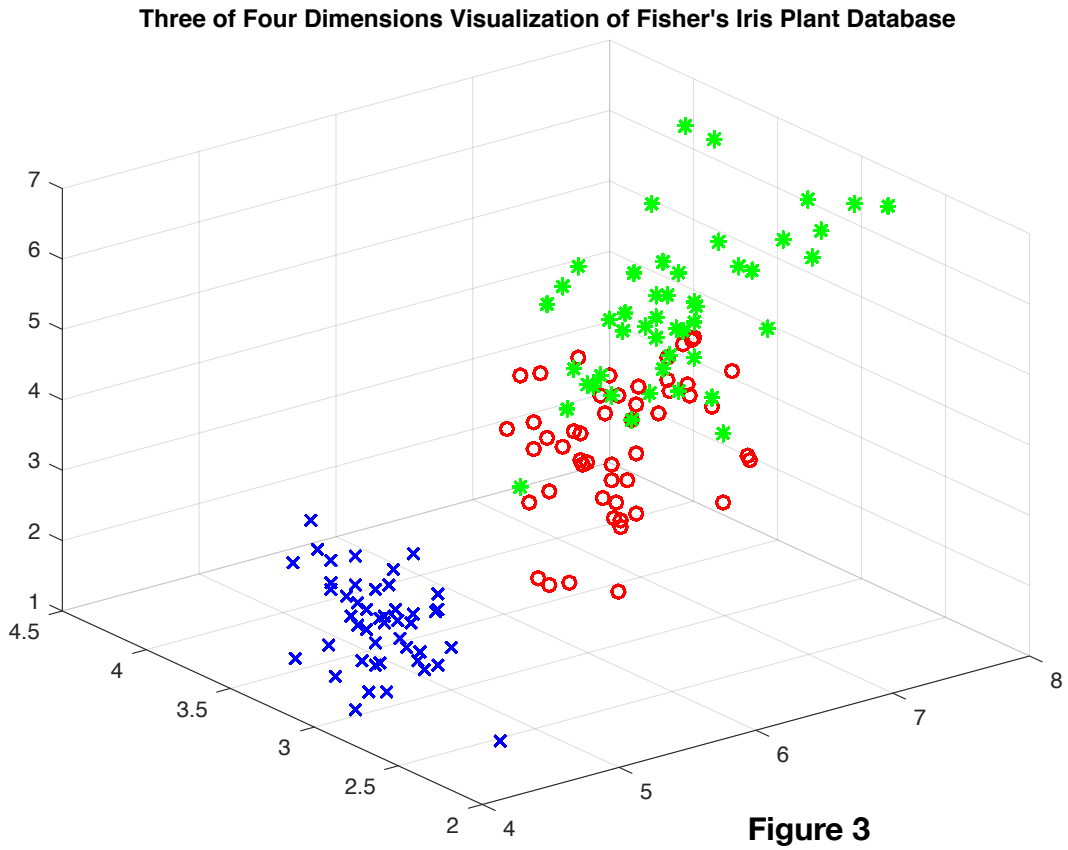

Figure 3