Jason Argo & Seth Lugibihl

SYS 411

Dr. White

03/07/19

In-Class Exercise: Iris Data

**Experiment Introduction**

The goal of this assignment is to utilize the Back Propagation Neural Network algorithm that creates the best classifier for a botanist to classify between Iris flowers. Our data has four input nodes representing the four independent variables of the flowers’ features utilized to classify between three types of flowers represented by three output nodes.

Due to runtime complexity and time constrains, we decided to limit the amount of iterations of our experiment seeking to be below 100,000. If we had more computational power and time, our experiment may have included a more comprehensive test of potential variables. Even with a limited amount of data test data, we were able to develop an effective classifier with randomized data sampling to provide statically back results to the validity of our system. If

**Back Propagation Experiment**

* Test Data Percent
  + We begin by shuffling our data list to ensure the data will be randomly sampled
  + Our experiment allows the user to specify a percentage of the data that will be randomly sampled without replacement to be the test data. The remaining data from our input list will serve as the training data.
  + For example, for 100 units of data specified at 20%, 20 units will be randomly popped from the list to serve as the test data while the remaining 80 will serve as the training data
* Number of Hidden Nodes
  + Our system takes in a list of values for the number of hidden nodes to be applied to the system
  + We initially looked into the recommended range of 2 – 9 test nodes for testing our system, but later settled on testing 2 – 5 hidden layer nodes due to initial experimentation with our system seeing diminishing returns beyond 5 hidden layers
* Learning Rate
  + Our system evaluates different learning rates we store in a list to test [.005, .009,.01, etc.]
  + The learning rate measures by how much should our system change with each iteration in order to reach an optimal solution.
* Number of Epochs
  + How many iterations do we want the system to run on our test data?
  + We selected the number of epochs for our system to run for each iteration
  + Note: This is one area where we may have done more testing (choosing higher values for epoch rate) if we had time. Though the system runs the risk of overfitting, we saw a very slight improvement from changing the epoch value from 100 to 150.
* Number of Random Test Data Samples
  + This variable controlled the number of samples we utilized for each run of a particular configuration of our system. The average results for test and training data for the number of iterations in the system was added to a running list to help us select the best system configuration.
  + This variable helped our system avoid possible fluke cases of perfect results for a configuration.
  + We decided to implement 10 as our standard because it is large enough for the system to avoid fluke cases in selection.

**Other Ideas (Note Implemented):**

* Normalization of the Data
  + During preliminary investigation, we found our system was performing very well without the need to normalize our data, so we decided to not implement this as part of our solution.
    - For other experiments, this may be a way to easily improve the efficiency of the system
* Cross Validation
  + This is an option to cross-check the system’s results by grabbing part of the test data as an intermediate cross validation set
  + Because of the systems effectiveness without cross validation (we are reaching over 95% accuracy on the test data without cross validation) we did not think this step was be necessary to get excellent results from our system
    - That being said, it may still have provided superior results and we would have certainly considered cross validation in other systems.

**Other Considerations:**

Elapsed Time

* + We calculated a run-time for running iterations of our system
  + This will help us understand how quickly our model was able to converge to a good solution and can be applied to calculate how long our system would take for larger runs and a greater number of system iterations.

**Test Configurations:**

We ran an initial test of our system and then a subsequently follow up test to see if changing the epoch value would improve our system performance.

**Test 1**

testDataPercents = [.1, .2, .3, .4]

HLNodes = [2, 3, 4, 5]

learningRates = [0.01, 0.015, 0.02, 0.025, 0.03, 0.035]

epochs = [100]

numSamples = 10

**Test 2**

testDataPercents = [.1, .2, .3, .4]

HLNodes = [2, 3, 4, 5]

learningRates = [0.01, 0.015, 0.02, 0.025, 0.03, 0.035]

epochs = [150]

numSamples = 10

**Test Results:**

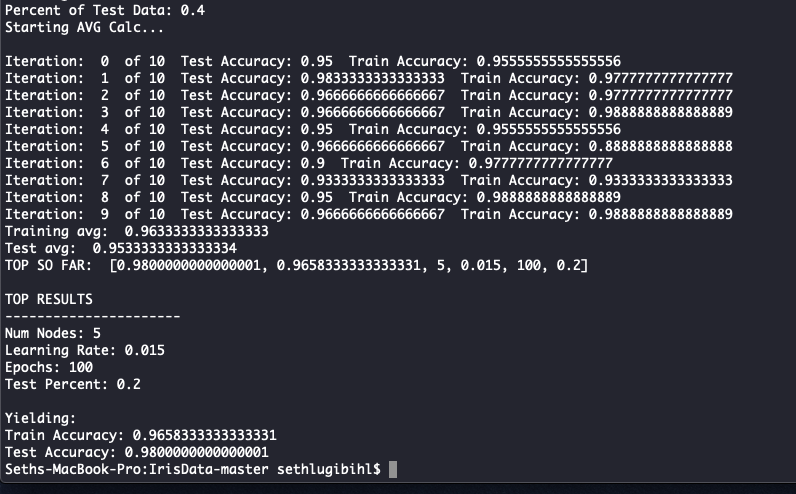
**Test 1)**

Average Training Accuracy: 0.9658

Average Testing Accuracy: 0.9800

Hidden Layer Nodes: 5, Learning Rate .015, Epochs: 100, Test Percent: 0.2

Time: ~1.5 hours



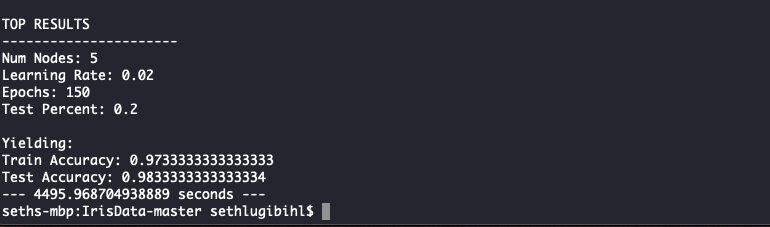
**Test 2)**

Average Training Accuracy: 0.9733

Average Testing Accuracy: 0.9833

Hidden Layer Nodes: 5, Learning Rate .02, Epochs: 150, Test Percent: 0.2

Time: ~2.5 hours



**Conclusions:**

Our experiment results showed that with 10 randomly sampled groups of testing and training data and 150 epochs of the system, we were got an average training data accuracy of 97.33% and test data accuracy of 98.33%. The superior configuration of our system included 5 hidden layer nodes, a learning rate of .02, and a test data percent of 20%. There was a slight variation in the learning rate and epoch values between our two experiments and we believe this is due to the interaction between the learning rate and the number of epochs of the system. The rest of the variables remained constant between these two systems.

Due to our random sampling and averaging of experiments, we feel confident that our system would run close to our test data effectiveness for its ability to help a botanist to classify Iris data.

**Future Considerations:**

If we had more time, we would continue to increase the number of epochs in our system until we see diminishing returns or signs of overfitting the data. We would also constrain and hone in other variables, such as the number of hidden layer nodes as our results consistently provided superior results for 5 hidden layer nodes compared to 2-4. Though we though we saw diminishing returns on hidden layer nodes greater than 5, we would likely try 6, 7, 8, and 9 hidden layer nodes once again because we kept getting the best result with 5 as the upper limit on configuration.