CSC 242 Project 4 Report

Runtao zhou

Siddhant Choudhary

Goutham Swaminathan

# Project Outline:

In this project, we implement linear classifiers, specifically a logistic classifier and a perceptron classifier. In addition, we also implement neural networks, specifically a multi-layer network with hidden units. In this report, we will include all graphs created from running our programs as well as the commands for running the programs themselves. For each graph, we outputted the results of our classifiers/neural networks into a text file which was then imported into RStudio to be plotted using the base plot() function.

Commands to run the project on terminal/console:

In order to run linear classifier for this project on terminal. You should type the following command:

java -cp “. /bin” learn.lc.core.PerceptronClassifier(or LogisticClassifier) [file name of data] [number of weight updates] [alpha learning rate ]

If you decide to use the numerical value for alpha, then directly enter the value you want to use for the third argument. Otherwise enter “decay” for decaying learning rate.

In order to run the neural network programs, we need to put the following commands:

Java -cp “. /bin” learn.nn.core.MultiLayerFeedForwardNeuralNetwork [program name] [epochs] [learning rate] [# of hidden units] [# of sets in k fold validation]

For program name, we only have two options: “IRIS” and “MNIST” (type without quotes).

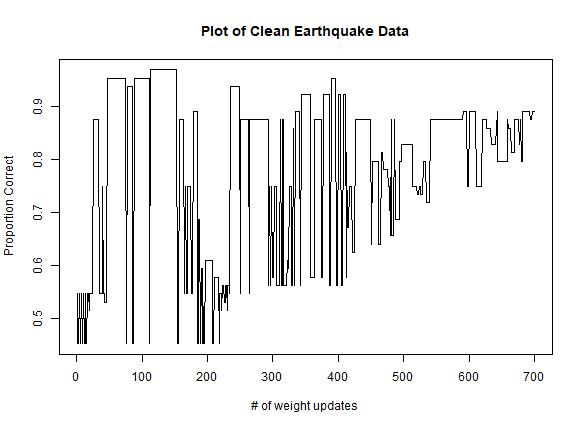
## Linear Classifiers

### Perceptron Classifier

For the earthquake dataset, we generated the corresponding graphs for both the perceptron classifier and the logistic classifier.

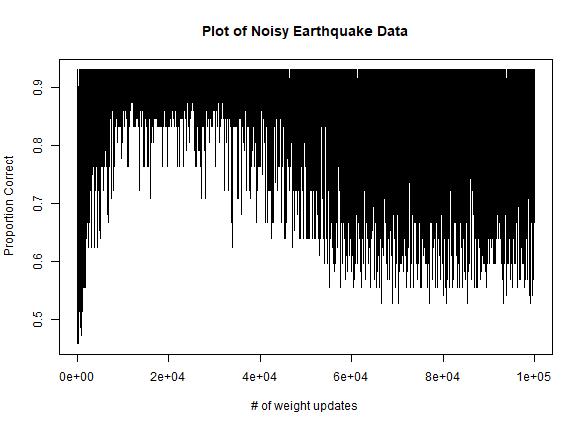
For the first graph plotting the results of the perceptron classifier on the clean data, we ran the following command argument and obtained the data for the following plot(Perceptron\_data.txt):

*“earthquake-clean.data.txt 700 decay”*



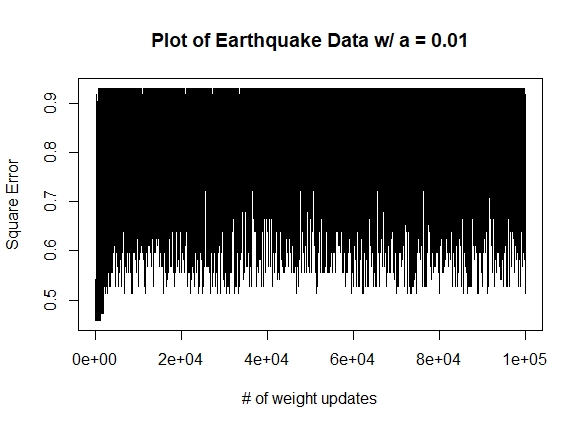
For the second graph plotting the results of the perceptron classifier on the noisy data, we ran the following command line argument and obtained the data for following plot(perceptron\_data.txt):

*“earthquake-noisy.data.txt 100000 decay”*



For the third graph plotting the results of the perceptron classifier on the noisy data using a learning rate schedule α(t) = 1000/(1000 + t), we ran the following command line argument and obtained the data for the following plot(perceptron\_data.txt):

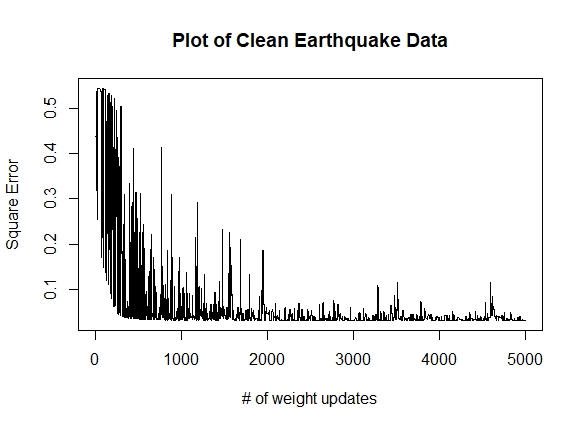
*“earthquake-noisy.data.txt 100000 0.01”*



### Logistic Classifier

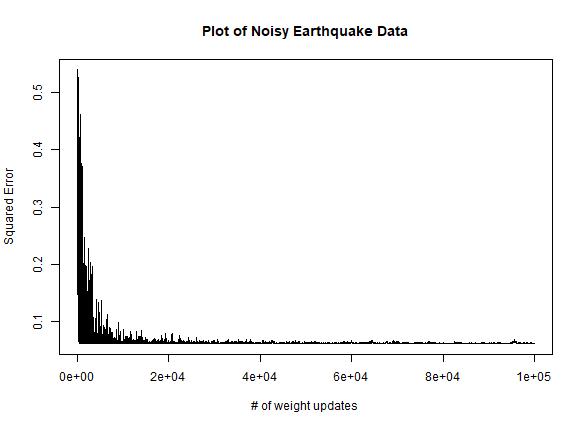
For the first graph plotting the results of the logistic classifier on the clean data, we ran the following command line argument and obtained data for the following plot (Logistic\_data.txt).

*“earthquake-clean.data.txt 700 decay”*



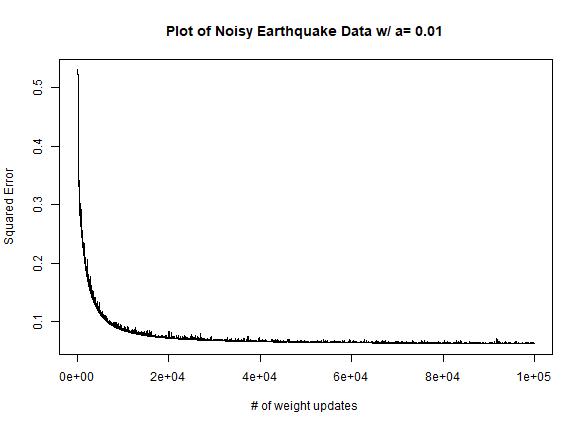
For the second graph plotting the results of the logistic classifier on the noisy data, we ran the following command line argument and obtained the data for the following plot (Logistic\_data.txt).

*“earthquake-noisy.data.txt 100000 decay”*



For the third graph plotting the results of the logistic classifier on the noisy data using a learning rate schedule α(t) = 1000/(1000 + t), we ran the following command line argument and obtained the data for the following plot (Logistic\_data.txt):

*“earthquake-noisy.data.txt 100000 0.01”*



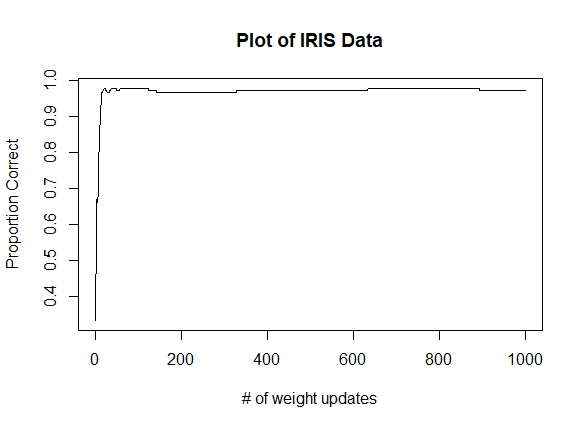
## Neural Networks

In order to run the neural network on either the IRIS dataset or the MNIST dataset, the first command line argument must be written as “IRIS” or “MNIST” to run the networks on the respective datasets. In addition, the last command line argument represents the value of k for k-fold cross validation.

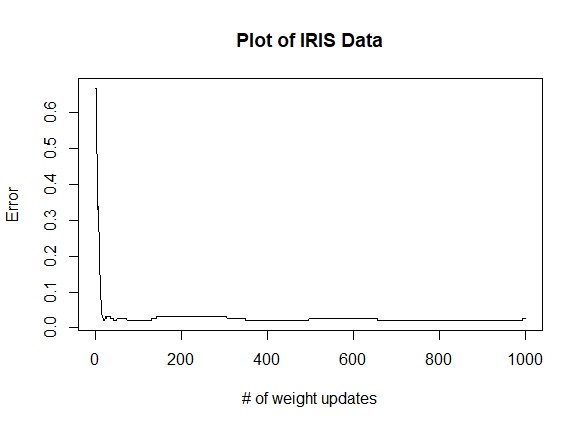
### IRIS Dataset

For the first graph plotting the results of the neural network on the IRIS dataset using 1000 epochs, a learning rate of 0.1, and 7 hidden layers, we ran the following command line argument and obtained the data for the following plot (NN\_data.txt):

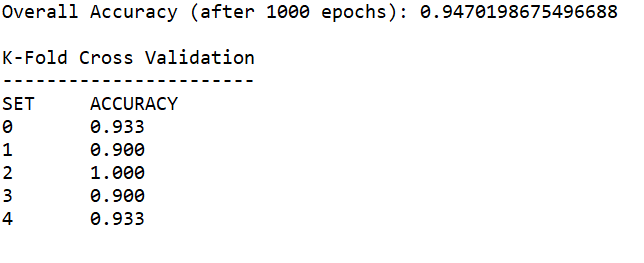
*“IRIS 1000 0.1 7 5”*



In addition, we plotted the error vs. the number of weight updates:



We also obtained the following values for overall accuracy and accuracy for each iteration of k-fold cross validation. We chose 5 for our value of k:

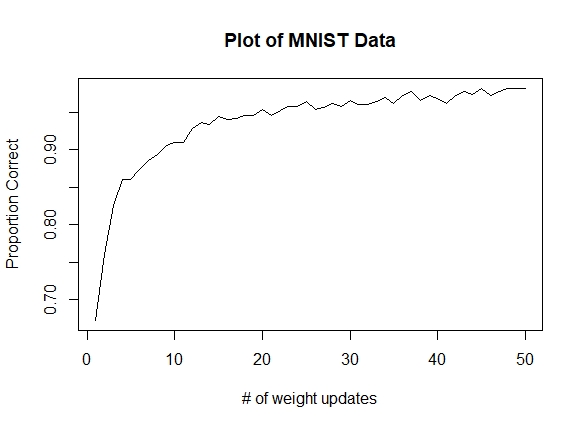


### MNIST Dataset

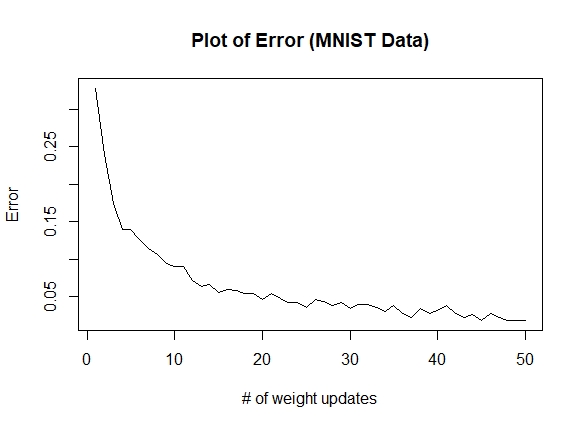
The neural network for the MNIST dataset had 785 input units (784 pixels + 1 bias), 300 hidden layers, and 10 output units.

For the first graph plotting the results of the neural network on the MNIST dataset using 50 epochs, a learning rate of 0.1, and 300 hidden layers, we ran the following command line argument and obtained the data for the following plot (NN\_data.txt):

*“MNIST 500 0.1 300 10”*



In addition, we plotted the error vs. the number of weight updates:



We also obtained the following values for overall accuracy and accuracy for each iteration of k-fold cross validation. We chose 10 for our value of k:

