Lane Henslee

Dr. MacLennan

CS425

November 19, 2019

**Project 4: Back Propagation**

**Project Description (10):**

The purpose of this project is to use back-propagation to build an artificial neural network on a data base from UCI that was donated in 1999. This is a multivariate data base with 57 attributes on 4601 instances of emails. Each email is then classified as spam or not.

**Pre-processing Steps (10):**

The initial format of the data is comma delimited with no missing data. The attribute columns are then normalized to have a mean of 0 and variance of 1. The data is then split into training data and test data. The dimensions, data sets, and number of neurons in each layer are stored in a class called ANN.

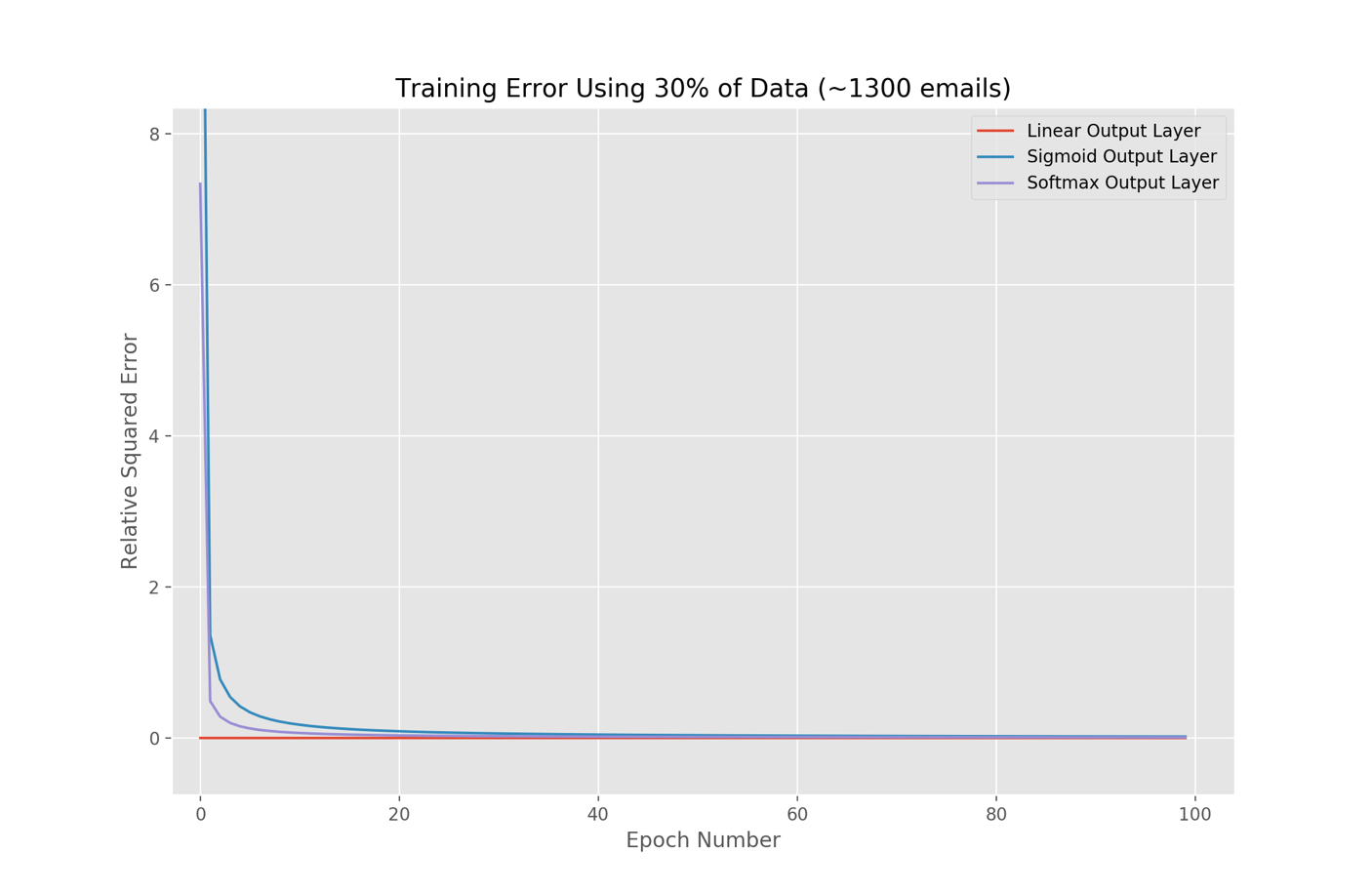
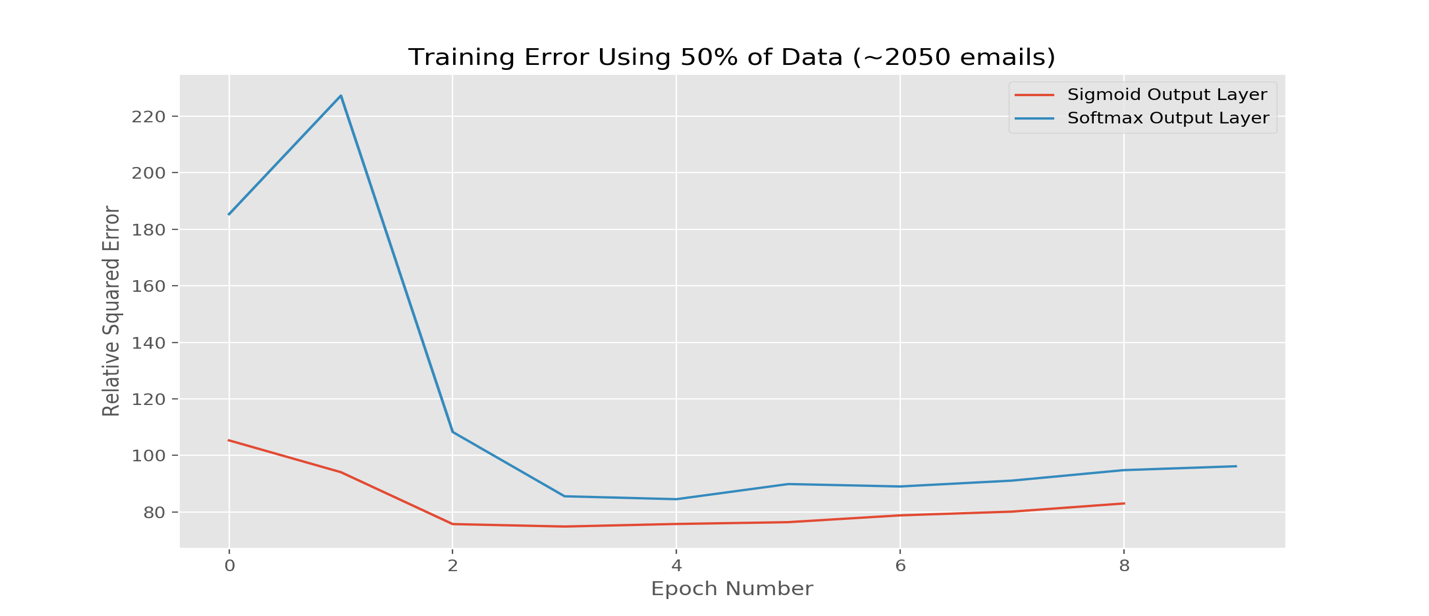
**Solution Description (20):**

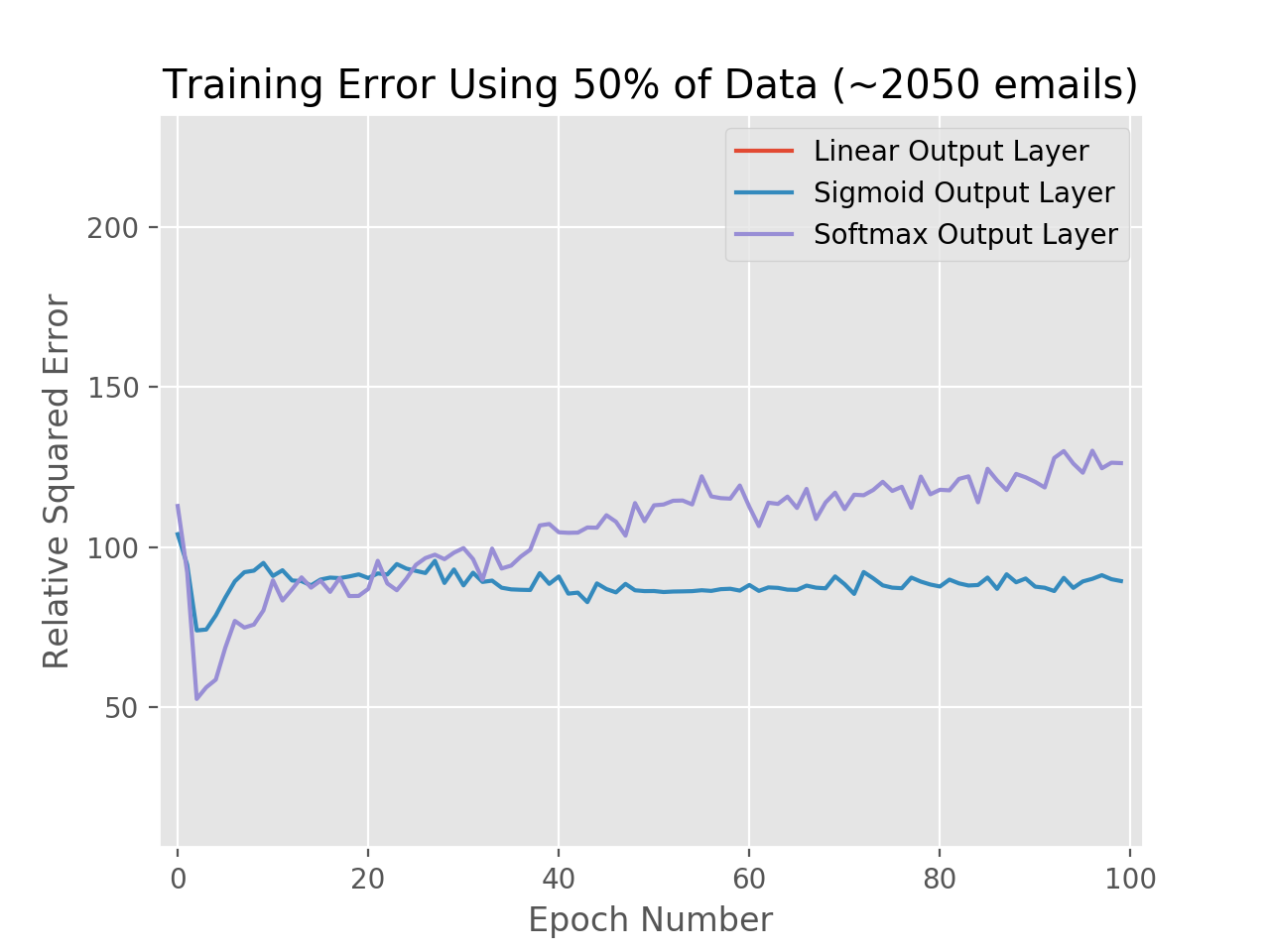
1. The first discriminant is the threshold function where the value becomes a 1 if the activation reaches the certain criteria. The threshold function makes the output layer linear.
2. The input layer can also “activate,” and forward-propagate with the sigmoid function, (output = 1 / (1 + e^(-activation))). output is the result of the sigmoid function and activation is the sum for perceptron.
3. Can also forward propagate with the softmax function, which takes two steps. Must sum together the exponential of all activations and divide the exponential of each instance by that sum.
4. Use the sigmoid transfer function to find the slope. (derivative = output \* (1.0 - output)).
5. Back propagate for error. (error = (expected - output) \* transfer\_derivative(output))
6. On each iteration, update weights with function, (weight = weight + learning\_rate \* error \* input)
7. Train and update the weight repeatedly for a certain number of epochs
8. Make prediction with trained artificial neural network by picking the argmax
9. Do dimension reduction with PCA

**Analysis (50):**

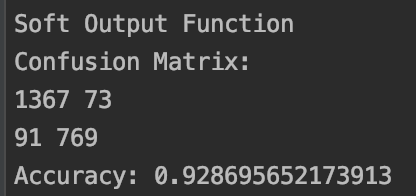
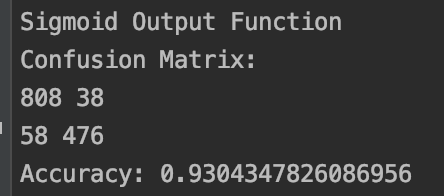
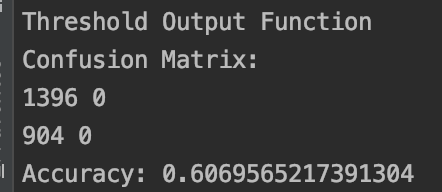
* (25) Provide results in legible format
* (25) Explain results in detail (not general summation)

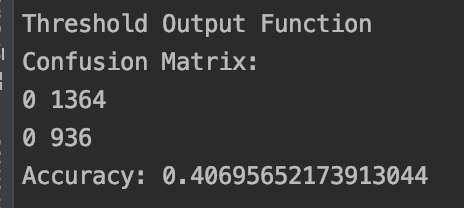
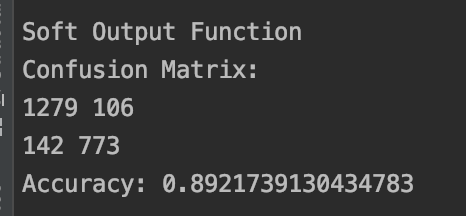
For the first graphical demonstration I gather 30% of the data samples and display the relative square error of each the linear/threshold function, the sigmoid function, and the softmax function over 100 epochs. On the second graphing display, I will increase the amount of training data and show how the linear function doesn’t learn as well as the other two. The XOR example comes to mind. Since this is just a simple hyper plane, when data reaches the wrong side the hyperplane, it cannot fix itself like the others. The learning rate is also .5 the entire time so I do risk over fitting the test data. Before the next graph, I will represent the 50% of the data (still showing the squared error), to show how the other two equations are superior.

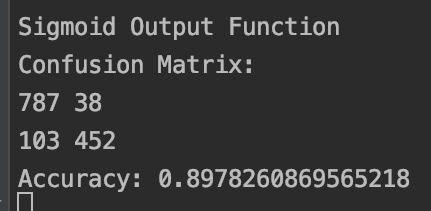
The graph shows that the linear threshold was never broken by the training data, so it never ran into any errors, the sigmoid function starts with the most error, and the softmax function starts out in the middle, and eventually they all almost reach 0 error. Now, I will increase the amount of training data and record the square error.

The linear output layer doesn’t even make it to the screen, because with all this data it slips up and cannot split the data apart linearly anymore. The sigmoid did better than the softmax over time, so next, I will make the epoch stop incrementing once the error increases 8 times consecutively, or the error reaches .009 to prevent over fitting. I will also implement an adaptive learning rate.

Lastly, I apply dimension reduction, because it lowers the error on the training data graph for both the sigmoid function and the soft max function. The confusion matrices for the sigmoid and softmax functions both present more accuracy than the threshold/linear function.

****

Here are the confusion matrices for each function when there is no dimension reduction.



**Discussion (10):**

The Sigmoid function and softmax functions are much more effective than the threshold function when building a multilayer perceptron. I made sure to shuffle the data every time I tested it too. Also, I learned how valuable dimension reduction is, because it saves more computing time than I would have though. Finally, it is possible to predict spam mail fairly accurately through artificial neural networks.