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):**

* (10) Include required equations

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

* (10) Explain all variables used in the equations

**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 300 epochs. This configuration explains why it is important to stop the epoch sooner, use an adaptive learning rate, and apply dimension reduction. 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.

**Discussion (10):**

* (10) General summation of results and report