Neural Network Lab 09 Write Up

In my TestNeuron class, I have three methods : main, readInput(), and train(). I first created my neuron by constructing it in the RELUNeuron class with the input amount of 500. This is because there will be 500 inputs to the neuron and 500 corresponding weights. This tests out the constructor. I then proceed to create a FileInputStream that creates my file “weights.dbl” that will write the best set of weights and bias to it. I then create an InputDataStream using the FileInputStream object as an argument. I set the best score to 100,000 and initialize the score to 0.0. The score represents the totalError calculated from all 100 training files each time train is called, and the bestScore is set extremely high so that the first scores will be the new bestScore. I then iterate 50,000 times and print the number for each time I do. In each iteration, the train() method is called with the neuron as its argument.

Train() has a string variable called “dbl” which is initialized as "TrainingData/NNTrainData00”. This is the string used to open the files because the only thing that differs from each is the last two numbers of it. So with each iteration based off of if the last numbers are from 1-9 or 10-99, the string is built differently to match each corresponding training file. There is also the score variable which is the weighted sum for each error calculated from each test file (100 of them). Train() also has a double array called inputSet of 501 values so that whatever readInput returns (an array of 501 values read from training files) is placed within the train method. Each time train() is called, the program reads through the 100 training files by a for loop that iterates 100 times. In each iteration readInput() is called where an array of 501 values are initialized for the 500 input values and the last as the expected output, and is returned – although it isn’t distinguished yet.

With each array returned from readInput, that array is then fed through the output method which takes in a double around and calculates the weighted sum and calls the activation method. Output() in RELUNeuron.java is also where the expected output value is initialized. Then after output() is called in TestNeuron, calculateError is called which calculates the absolute value of expected output - output. That error is returned and added to “score”. After the 100 files have been iterated through and the neuron has been fed through the output, the score is returned to main. In main the for loop of 50,000 iterations takes place. The for loop sets the score in main to the score returned from train(). It checks if the current score is less than the best score. In the first iteration, it will be because bestScore is set to an extremely high number on purpose. If the

current score is less than the previous best score, it becomes the new best score and “NEW BEST” is printed. Then the write method is tested by writing the current set of weights and bias that the neuron had when the new best score was produced to a DataOutputStream called “weights.dbl”. Each time a new best is found, the file is overwritten based on how write() is written in RELUNeuron.java. If the best score is not better (less than the best score) then we call tweak() on the neuron in the RELUNeuron class. It tweaks the weights and biases randomly. I messed around with things a bit before I found that it works the best, and I don’t exactly know why.

Training the neuron altogether takes the for loop in main, the train() method, readInput method, and of course uses output() and calculateError(). When iterating through this program numerous amounts of times, I found that the score drops initially very well and once it gets to around 15.2-5, it stops dropping drastically and gets stuck there for a while. It takes a very long to get the next best score, and it decreases very minimally. After iterating 50,000 times, I’ve found the lowest I’ve got was around 14.93 with a bunch of decimals at the end. Each time I do I get a number very close to that. My program takes a while to run, and I am unsure how to lower the score more drastically. However, it does technically get better and therefore does what a neuron is supposed to do. I test read() by reading the supposed last set that was written to weights.dbl. This is because write() is only called when a new best score is found, and it writes the weights and bias the neuron had when it got that best score (it’s overwritten each time).

After the 50,000 iterations, read() is called. So whatever the last best set was, read() is able to read the values last written. It was difficult to test it out. However, I made sure the values were the same by decreasing my for loop in main to 50 and 500. I have commented out print statements in read() that print out each of the weights, and I have commented out lines including print statements in write() that print out the weights as well. After the 50 or 500 iterations, I also had to print out the 500 weights of what write() was writing to weights.db,l, and what read() was reading from - from weights.dbl. I looked at the random indices of the same number in each and made sure those numbers were the same, and from my tests, they were. This is how I tested out all the methods in my program. I also added a lot of comments and descriptions in case of any confusion. There are also the commented-out lines I used to do what I just said to test if my write() and read() methods work correctly.