Agathoklis Georgiou

CSC 378

Midterm Project Documentation

Table of Contents

1. Instructions of use.
2. Description of the system.
3. Ideas for Enhancement.
4. Instructions of Use

The project was designed in the Net Beans IDE under Windows 7. This free IDE can be found here: <http://netbeans.org/>.

To input the user parameters I accept 2 input files “activation.txt” which contains the input activations and target output, and “parameters.txt” which contains the number of examples, input and hidden units, maximum epochs, learning rate, and error margin. All input is dynamically handled, yet it must be entered correctly.

Input in “activation.txt” must be in the format shown below:

0.222222,0.625000,0.067797,0.041667,0.00

0.166667,0.416667,0.067797,0.041667,0.00

0.111111,0.500000,0.050847,0.041667,0.00

0.083333,0.458333,0.084746,0.041667,0.00

0.194444,0.666667,0.067797,0.041667,0.00

0.305556,0.791667,0.118644,0.125000,0.00

0.083333,0.583333,0.067797,0.083333,0.00

0.194444,0.583333,0.084746,0.041667,0.00

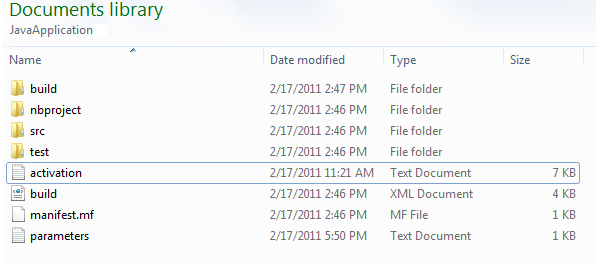
0.027778,0.375000,0.067797,0.041667,0.00

Where each comma separated value is an input activation, and the last value is the target class.

Input in “parameters.txt” must be in the format shown below:

150,4,10,200,0.1,0.05

Where each coma separated value applies to the number of examples, input and hidden units, maximum epochs, learning rate, and error margin, respectively. No spaces after commas are acceptable. Both input files must be located in the Net Beans default project folder just as shown below

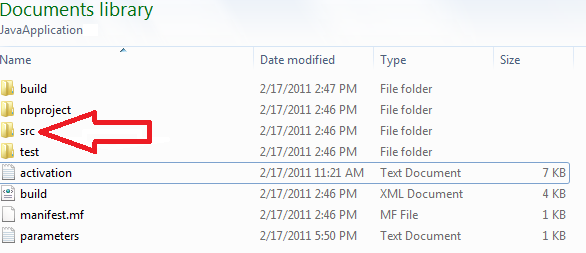


After Creating a new Net beans project, the Directory to its folder would be at:

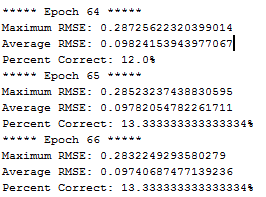
C:\Users\”username”\Documents\NetBeansProjects\JavaApplication1

The Java files included in my compressed folder should be placed in the src folder located in:

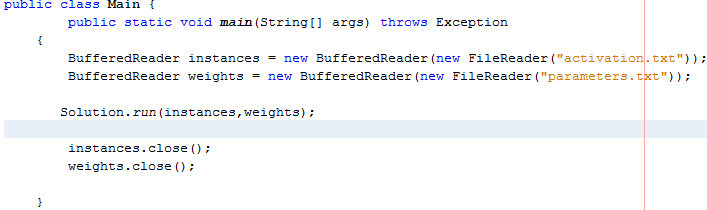
C:\Users\”username”\Documents\NetBeansProjects\JavaApplication1\src



Output will be shown in the Net Beans console just like below:



When everything has been set, the program is ready to run. To change the names of input files, the text file names can be changed in the main class like shown below:

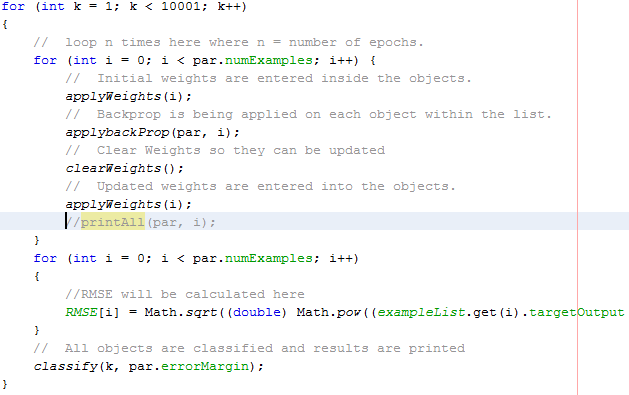


2. System Description

1. I create a custom parameter object which gets all the arguments from the”parameters.txt” This object will be used throughout the whole class to provide its arguments to other methods.
2. I then print the initial given parameters and proceed to create an exampleObj.

This object is essentially a custom object which simulates a whole neural network. It contains an output, and a list of input and hidden nodes. The input and hidden nodes are also customized objects which contain their activation and know their weights to other nodes.

1. Then I get the initial weights from the input file to an array which I will use to populate all my object nodes with their weights.
2. Then a loop will call methods to apply the backprop algorithm and provide the answers. The weights are first applied the example, backprop is applied to the weights array, then the weights are transferred to the example object. This procedure will repeat for each example until one epoch is complete. Then in the training phase, the Max, Avg RMSE and percent Correct are calculated and shown on the console. This code section is shown below:



Some code was left out on the right part of the screen, but can be found in Solution.java.

Also, a printAll(…) method is available for the user to use for the contents of the example object weights and activations to be printed on the screen. I found it quite useful while debugging.

3. Ideas for Enhancement

In this Project I decided to take advantage of Java’s strong object oriented nature. Thus I created a lot of objects which simulate input and output Nodes and even a whole neural network. What is great is the flexibility that provided for me while designing this application. There is a cost though, and that is complexity. So, several times I had to search very deep in my data structure to find the information I was looking for, and that took a lot of time. Also correcting errors was hard due to the complexity.

I believe that I could have done more refactoring and delegation. My idea for custom objects was quite successful, thus I should have put more content into the object classes, and less in the Solution class. Also, I could have simplified some of the logic to avoid errors and debugging.

After running 10,000 epochs on some of the files provided, the application seems to be getting the correct results.

Designing this application was a great experience and I am looking forward to implementing more neural network algorithms in the future.