Computational Intelligence NNet Homework

General

You've installed the LiveCD and booted it up. You're now ready to experiment with neural networks.

The environment is GUI based but I think it's easier to do a bit of command line things. See Appendix for a short Linux tutorial.

First let's navigate to the appropriate directory.

- Double click on the terminal icon on the upper left of the screen.
- Enter a `ls` command[1].
- Enter `cd ucsc_ci` and do an `ls`. You should see the `irisfann` directory (in blue).
- Do a `cd irisfann`
- You should now be in the correct directory. You can verify this by entering, `pwd`.
- Do an `ls`. This should show the programs you need to run, `goscale, gotrain, gouse` in green, indicating that they are executable.

1 Iris Problem

The goal is to predict the species of Iris flower from four pieces of data; sepal width, sepal height, petal width, and petal height. There are three possible species a particular plant could belong to.

As we discussed in class you are given a data file containing the canonical Iris data. Using this data you will run two programs to produce a trained neural network, one program to scale the data and one to actually train the network. Once you have a trained neural network you must test it on unseen data and evaluate the results.

So, now that you are in the correct directory, you can train and run your first neural network. Table 1 describes the programs you must run.

Type `./goscale` to scale the data. This produces a scaled data file from the raw data.

Type `./gotrain` to train the network. There are default settings allowing it to run. Later you will edit this program to change the network structure and/or the parameters to optimize the performance of the network.

Type `./gouse` This will test the network on unseen data and show the results.

The display shows the correct solution and the neural net solution. A mean square error is also provided to help gauge your results.

The next step is to improve the network's performance.

You will do this by editing the train program as shown,

```
gedit Train.hs
```

[1] See Appendix for Linux command summary

Table 1: Command Summary

./goscale	Scales the raw data. Neural networks like scaled (normalized between -1 and +1) data. Both the inputs and outputs are scaled. Thus the output value of -1 would indicate species one, a value of zero would indicate species number two, and a value of +1 would indicate species number three.
./gotrain	Trains the neural network. This program is the one you will edit to change the learning parameters and network structure, and by doing so improve it's accuracy
./gouse	Tests the trained neural network (feed forward). Giving it data it has never seen in training it will used the trained neural network to predict the species of a flower

Note that `gedit` is a 'standard' editor similar to notepad in Windows. I won't explain it's operation here.

Near the top of the file you will see, fannDef = [4, 8, 1]

Change that to, fannDef = [4, 8, 4, 1]

This adds one hidden layer with 4 neurons.

Save the changes.

Exit `gedit`

Now when you type, ./gotrain It will use your new modified file, and you can see the new results both in the training (which you just ran) and also in the usage (when you type ./gouse).

Inside Train.hs you may play with the network topology (described above), the epoch count (change from 2000), the desired error, learning rate, and momentum. The learning rate just says how much of the error will be applied to correct the synaptic weights; bigger (1 max) is a faster learning rate. The momentum just says how fast can we change weights from their present value to a new value; bigger (1 max) means that weights can change faster.

Remember – is a Haskell comment.

Now just document the results of your experimentation.

Appendix, Linux Tutorial

`ls` List directory. The equivalent of 'dir' in Windows. Directories and files will be listed. Directories are blue in color.

`cd` ***directory name*** Change directory. Move inside the indicated directory. The command `cd ..` will move you up one directory level. Entering `cd` with no operand will take

you back up to your `home` directory, the highest level of directory you have access to, `/home/poliquin`. This can be convenient shortcut.

`pwd` Displays the current directory, the directory you are currently in. E.g. `/home/poliquin`

`./command` Execute a command present in the current directory. E.g. `./gotrain`

Tab Completion To simplify the typing of commands and filenames, tab completion may be used. Simply type a few characters and hit the `tab` key. The computer will fill in the name if there is a unique choice. If no unique choice exists hitting the `tab` key twice and it will show you options. This feature is incredibly useful.