CS475 - Assignment 2 (Multilayer Neural Network)

REWARD: 200 points
DUE: Sat Feb 26 at 11PM PT
TEST DATA: testDataA2.tar or testDataA2.zip
LIBRARY: mat.tar, mat.zip

---

1 The Task

Let's do multilayer neural networks from chapter 4 in the book.

Build a neural network in C/C++ with a single hidden layer as we saw in class. It should solve a problem that consists of **N** inputs and **M** outputs. Your program will be called **nn**. Use the matrix library to make your life easier. Look at the NN notes in the references section. The batch two layer neural network is geared toward a matrix solution. And it is much easier for me to grade. WARNING: this assignment is harder than it looks.

2 The Training

Your programs will read in a matrix of data from standard input that will be used for both training data and test data. We will deal with training/validation/test data later. You should then scale your input to be between 0 and 1. Your program can then train as much as you believe is needed. For this assignment use all the training data as a block. The input file looks something like:

```
NumInputs
NumRows NumCols
row1
row2
...
lastrow
```

The first number, NumInputs, is the number of features or inputs. The training and test data are identical and look like a matrix suitable for the read() in the matrix library. All elements in a row after the first NumInputs elements are the expected targets in the training data. Note that the NumCols in the training matrix is always greater than NumInputs.

The number of hidden nodes will be read in on the command line (using argc/argv). It will read in the test data and then use the trained network on the test data itself to derive the outputs.

An example call of the program from the test script:

```
nn 3 < irisData.in
```

where 3 is the number of hidden nodes. See the test data for example output.

Your network is a two layer network with the classic sigmoid transfer function:

$$\frac{1}{1.0 + \exp^{-4.0*x}}$$

This is some example input that will test the two input logical function with three outputs (and, xor, and implication). You can see there are two inputs and a matrix for training.

```
2
4 5
0 0 0 0 1
0 1 0 1 1
1 0 0 1 0
1 1 1 0 1
```

For the Iris Problem there are 3 species or classes. The Iris Problem is a very famous data set (see page 93 in the book). It is so famous it has its own Wikipedia page!! (See "Iris flower data set") In the case of the iris data sets you'll see the training is conveniently in "1 of 3" form. The test data testDataA2.tar shows more examples of the input format. For this assignment just do the standard sigmoid and solve for the 3 expected outputs. That is, don't use the fact in learning that the encoding is "1 of 3".

3 Output Statistics

After training and printing out the results for the test data (same as training in this case).

In the output statistics we will assume the output is "1 of n". With this assumption a confusion matrix is an appropriate statistical measure. Show the confusion matrix for the test run. A column represents a predicted value and a row a given target value. There is an argMaxRow method in Matrix that is useful for picking which species by picking

the largest value. I also used get() and inc() operators in a loop. But the confusion matrix is very simple to calculate with the help of the matrix library. So the learning part before the output statistics is just learning target vector for each input vector. The statistical part assumes the output is actually "1 of n".

Your code must compile and run on the class unix machine. If it does not compile or fails to run (e.g. gets seg faults) or in other ways produces no output that is a very serious fault and will result in a very poor score. In 4xx/5xx CS classes your code should at least run and produce reasonable output.

Grading will be based on subjectively looking like my output. Since these programs are stochastic I will look that the format of your output matches and the values seem reasonable. Test suite will do a side by side comparison using the UNIX sdiff tool. See information on the class page about testing.

Submission

Tar up all the code necessary along with a ¡a href=make.html¿ makefile to build the program named **nn** that reads the sample data from stdin as described above. Homework should be submitted as an **uncompressed** tar file to the homework submission page linked from the class web page. You can submit as many times as you like. **The LAST file you submit BEFORE the deadline will be the one graded.** For all submissions you will receive email at your uidaho.edu mail address giving you some automated feedback on the unpacking and compiling and running of code and possibly some other things that can be autotested.

Have fun.