CS475 - Assignment 1 (Using the Matrix Library to Build a Perceptron Model)

REWARD: 150 points
DUE: Thu Feb 10 at 11PM PT
TEST DATA: testDataA1.tar or testDataA1.zip
LIBRARY: mat.tar, mat.zip

---

1 The Task

The perceptron is a fundamental building block of a neural network. So let's play with it before moving on to multiple layer neural networks.

Build a perceptron network in C/C++ (single 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**. You may use the following code but all the rest needs to be written by you (do not copy code!! See class policy.): mat.tar , mat.zip These are matrix and random number routines discussed in class.

2 The Training

Your program will read in training data from standard input. You should then scale your input to be between 0 and 1 (See Matrix Library). Your program can then train as much as you believe is needed and then print the W matrix. It will then read in test data. For this assignment use all the training data and then use the test data. Do not train on the test data. We will deal with cross validation later. The input file looks like:

```
#inputs
#rows #cols
row1
row2
...
lastrow
#rows #cols
row1
row2
```

```
...
lastrow
```

First number is the number of features or inputs. Let's call it N. The remaining input is the training and test data. Each look just like a matrix that can be read using the provided Matrix Library. For both matrices N is the number of inputs (size of $\vec{x}$) and the remaining elements in the rows are the expected outputs. Test data doesn't normally come with expected outputs but we will for this assignment so you can see how well your perceptron is doing on the test data. Your program will read in the training data and then train. Your network is a single layer perceptron network with a transfer function:

$$\frac{1}{1 + e^{-\text{slope}*x}}$$

where slope is a parameter that you pick. Then it will read in the test data on stdin and use the trained network on the test data to derive the correct outputs. I have the following constants in my program:

```
const double eta
const double slope
const double spread
const double bias
const int numPasses
```

You should pick values that work well for you. Your program will have a time limit of 10 seconds to cover all of the tests.

This is some example test input (the two input logical or):

```
2
4 3
0 0 0
0 1 1
1 0 1
1 1 1
4 3
0 0 0
0 1 1
1 0 1
1 1 1
```

(The test data in testDataA1 shows more examples of the input and output format.)

Your program will print out:

1. the normalized input training cases and bias column as a matrix. See examples in the test data file.
2. The expected output from the training matrix.
3. the normalized input test cases and bias column as a matrix.
4. the expected output for the test cases
5. the **unnormalized** input test cases and bias column and result of using you perceptron on the test case and the diff between the expected result and result from your perceptron all as a single matrix.

Please try to match the output format exactly. For instance if I print 1.00 you should not print 1 or 1.0000. Using the Matrix Library should get you the right output format by default.

You may use my random number generator and matrix objects or write your own. If you do, then be sure to include the Matrix Library and any other code you need to build in your tar file! Do not use any other prepackaged software. Again, please try to match my output format.

Your code must compile and run on the test machine (a Linux 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 very poor score. In 4xx/5xx CS classes your code should at least run and produce reasonable gradable output.

Grading will be based on matching my output. Since these programs are stochastic, I will look that the format of your output matches and the values seem reasonable. If your output is wildly different in format I won't spend time trying to figure out what number is what. I will count it as a failure.

Test suite will do a side by side comparison using the standard UNIX sdiff tool. See information on the class page about testing.

An important note about compiling. As our programs get more complex they will take more time. It is best you compile all your C++ with at least the -O3 option to get some seriously optimized code. This:

```
CXXFLAGS=-O3 -Wall   # optimize and complain
```

in your makefile will help a lot. All assignments have a generous time limit but not a crazy large time limit.

3 Submission

Tar up **all** the code necessary along with a makefile to build the program named **nn** that reads the sample data from stdin as described above. Homework should be submitted as **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.