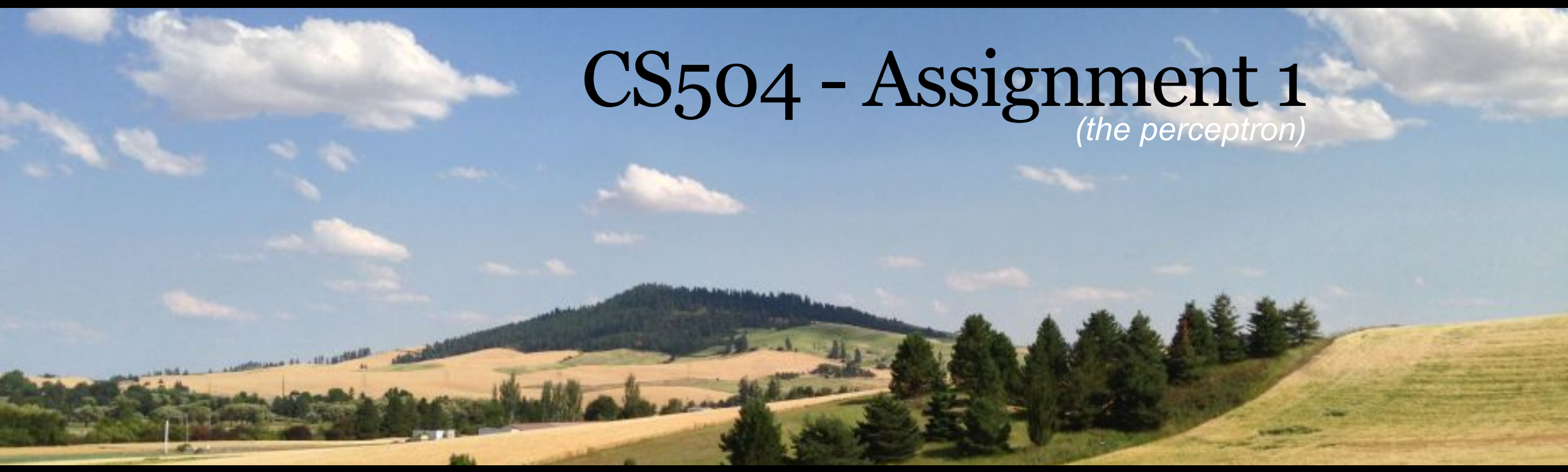

CS504 - Assignment 1

(the perceptron)

REWARD: 150 points

DUE: Thur Feb 9 at 5PM PT

TEST DATA: testDataA1.tar

---

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!!): matRand.tar, matRand.zip These are matrix and random number routines discussed in class. A sample test program is in mat.cpp and can be uncommented.

the training

Your program will read in training data from standard input. You should then scale your input to be between 0 and 1. 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
```

The training and test data each look like a matrix specification. First number is the number of features or inputs. All elements in a row after the first #inputs elements are the expected outputs in the training data. The #rows in the training matrix is greater than #inputs. The #rows in the second matrix is the same as #inputs. Your program will read in the training data and then train. Your network is a single layer perceptron network with a transfer function **g(x)** that is 1 if x>.5 and 0 otherwise. Then it will read in the test data and use the trained network on the test data to derive the correct outputs.

This is some example input will test the two input logical or:

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

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

(The test data testDataA1.tar shows more examples of the input format.)

Your program will then print exactly:

```
BEGIN TESTING
```

Then for each test case it will print out **on one line** the unscaled input test case followed output using your learned W matrix. There are some single line printing routines in the matrix library that you might find helpful.

The test script will compare everything after the "BEGIN TESTING" mark with the expected results.

You may use my random number generator and matrix objects or write your own. If you do then include the matRand library be sure it is in your tar file! Do not use any other prepackaged software.

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 poor score. In 4xx/5xx CS classes your code should at least run and produce reasonable 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. 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 makefile to build the program named **nn** that reads the sample data from stdin as described above. Submit a tar for each assignment. The tar can be the same or different depending on what you decide to do. 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.