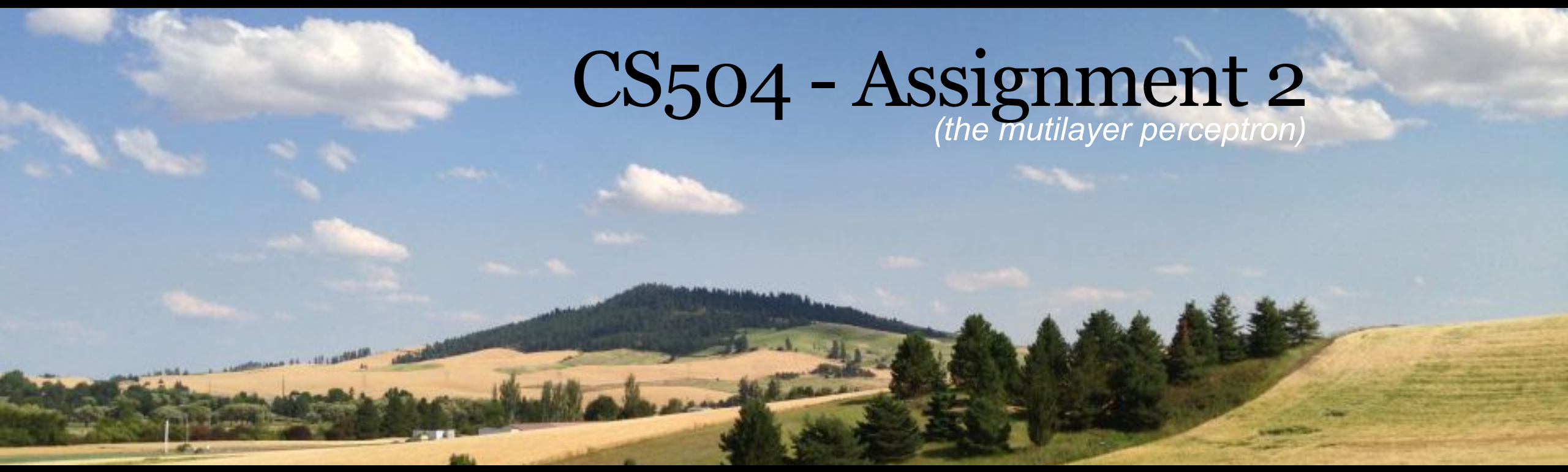

# CS504 - Assignment 2

*(the mutilayer perceptron)*



REWARD: 220 points

DUE: Thu Feb 23 at 5PM PT

TEST DATA: testDataA2.tar or testDataA2.zip

---

## the task

Now build a multilayer neural network in C/C++ using the algorithm on page 78 in your book. Use a single hidden layer. Your network will classify iris (the flower) species by looking at four lengths of the flower. This problem is not as easy. But doable for a multilayer network. For each of the four measures there is a classification 0, 1, or 2 which is the species, however, I have converted this to three channels of output for you. Build a network with 4 inputs nodes and 3 output nodes, one node for each possible classification. Assume a bias node of -1. Use the **classic sigmoid function**, 1/(1+exp(-4 * slope *x)), with a slope of your choice. Normalize the input appropriately as discussed in class. Don't forget to normalize your training data to the same scale as the training data. See the matrix library **normalizeCols** methods which provide this capability.

## the training

Your program will read in training data from standard input (see below). Your program can then train as much as you believe is needed within a 30 sec time limit on the testing machine. **For this assignment do your training in batch mode.**

It will then read in test data. For this assignment use all the training data and then use the test data to see how well your net was trained. Do not train on the test data. The input file format looks like:

```
#inputs
#numberOfHiddenNodes
#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. The second is the number of hidden nodes. Then comes the dimensions of the training data: number of rows and columns. The #cols in the training data matrix is greater than #inputs. The #cols in the test data matrix is the same as #inputs. Your program will read in the training data and then train. Then it will read in the test data.

Your program will then print exactly:

```
BEGIN TESTING
```

Then for each test case it will print out **on one line** the original unnormalized input test case

followed output using your learned W and V matrices and given number of hidden nodes.

Create and print a confusion matrix for the training data.

Finally, print the word "DIST:" followed by the Euclidean distance between the predicted and target values for the training data. This would be the sqrt(output.dist2(target)) using the matrix library.

Do not use any prepackaged software other than the supplied matrix library.

## implementation

My Matematica implementation of the problem is here. Except where noted (such as transpose and dot operators) the matrix operators modify the object through which the call was made. `t.add(u)` will modify `t`. Remember to watch out for the resizing of deltah to trim off the bias column. Adding a column can be done by creating a wider array and inserting and/or setting the information in the matrix. Removing the last column can be done with the **narrow** operator.

## submission

Tar up the NN with a makefile to build the program nn that reads from stdin as described above. Submit a tar for the assignment. The tar can be the same or different depending on what you decide to do. Homework will 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.