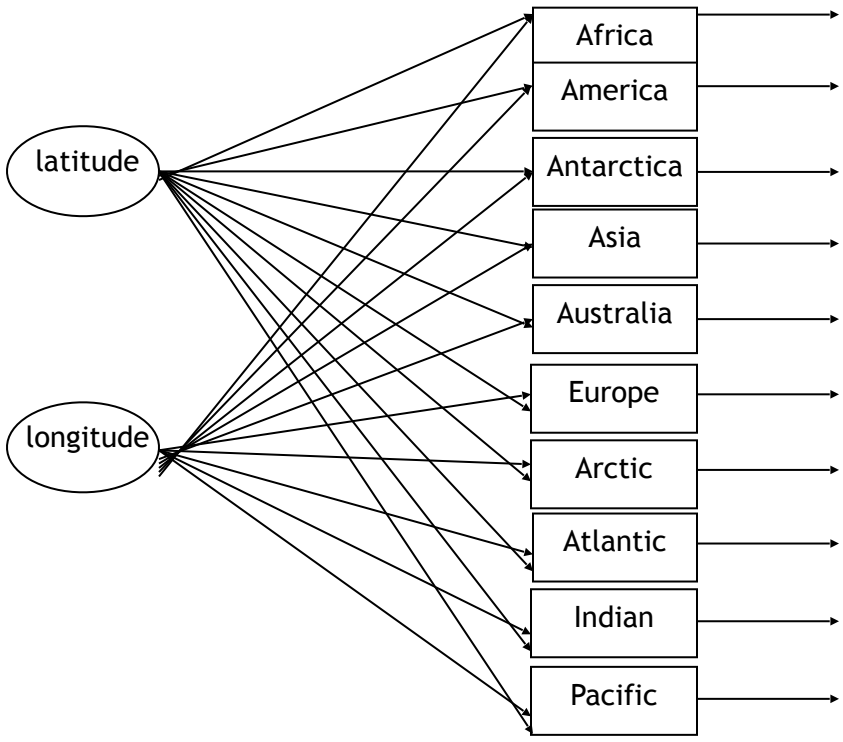
Project 3 – Neural Networks

Goal: To gain experience with implementing an artificial neural network and with the design choices that accompany using one.

Description: You're going to create a neural network to learn to recognize which continent (or ocean, if the land doesn't formally belong to a continent) a particular pair of latitude and longitude coordinates correspond to. You will use perceptron learning to train your network. If you wish to create a multi-layer neural network, then you are certainly welcome to do so – extra credit will be available if your multi-layer network consistently outperforms a perceptron.

The inputs to each node will be a latitude measurement (a real number in the range -90 to 90) and a longitude (a real number in the range -180 to 180) – you are free to normalize these values. The network will have 10 output nodes to classify the location as belonging to "Africa", "America", "Antarctica", "Asia", "Australia", "Europe", "Arctic", "Atlantic", "Indian" or "Pacific". For example, if the coordinates correspond to Asia, the "Asia" node should output 1 and all others should output 0.



You have been given a sample set of training data (nnTrainData.txt) and a separate set of testing data (nnTestData.txt). **Hint:** when debugging your code, you must create a new and much simpler training set, e.g. 4 examples all from Africa and no examples from anywhere else, or 2 examples from 2 different continents and none from anywhere else. The smaller training files will allow you to trace your code by hand and make sure that all updates are being performed correctly before you move on to the full training set. Similarly, you might want to initialize all of your weights to a constant like 0 or 1 (rather than a random number) when you're debugging to make it easier to figure out whether they are updating appropriately. You can then add in the randomization code once things have been debugged.

Your program should provide a simple menu interface to do at least the following:

- Open a set of training data (whose filename is supplied by the user) and perform a standard neural network learning algorithm to adjust the weights. Any settings that are adjustable, e.g. initial learning rate, number of epochs, etc. should be adjustable by using a set of clearly named global constants that the grader can easily modify.
- Store the currently learned weights into a file, using whatever format you choose.
- Either load in the initial weights for the neural network from a file or initialize them randomly.
- Open a set of testing data and run the samples through the network. Compute the following statistics and display them at the end of the test set:
    - What percentage of the examples in the testing data set were perfectly classified: i.e. only the correct neuron and none of the others fired.
    - What percentage of the examples in the testing data set caused multiple neurons to fire.
    - What percentage of the examples in the testing data set caused zero neurons to fire.
    - For each individual output neuron, calculate the percentage of times that it (a) fired correctly, (b) fired when it shouldn't have, and (c) failed to fire when it should have.

It's your choice what threshold value you like for the neurons as well as whether to include a bias input.

Benchmark: To get full credit for the assignment, you should play around with the available parameters (e.g. learning rate, number of epochs, etc.) until you are getting strong performance out of your network. When you submit your code, the parameters should have been left in that configuration and your best set of learned weights should have been stored into a text file. Your code will be tested on a different test file from the one provided to you and your success rate will be compared to some benchmark code that the grader have written. Here is the performance that the benchmark code achieves on the training and testing data you have been provided with:

```
Neuron: Africa
   Correct: 79.08%
   True Positives: 2.08%
   True Negatives: 77.00%
   False Positives: 20.32%
   False Negatives: 0.60%
Neuron: America
   Correct: 99.15%
   True Positives: 20.52%
   True Negatives: 78.63%
   False Positives: 0.15%
   False Negatives: 0.70%
Neuron: Antarctica
   Correct: 100.00%
   True Positives: 0.00%
   True Negatives: 100.00%
   False Positives: 0.00%
   False Negatives: 0.00%
Neuron: Asia
   Correct: 94.39%
   True Positives: 25.26%
   True Negatives: 69.13%
   False Positives: 2.88%
   False Negatives: 2.73%
Neuron: Australia
```

```
   Correct: 99.71%
   True Positives: 1.07%
   True Negatives: 98.64%
   False Positives: 0.12%
   False Negatives: 0.17%
Neuron: Europe
   Correct: 87.18%
   True Positives: 38.74%
   True Negatives: 48.43%
   False Positives: 5.50%
   False Negatives: 7.32%
Neuron: Arctic
   Correct: 100.00%
   True Positives: 0.00%
   True Negatives: 100.00%
   False Positives: 0.00%
   False Negatives: 0.00%
Neuron: Atlantic
   Correct: 99.80%
   True Positives: 0.00%
   True Negatives: 99.80%
   False Positives: 0.06%
   False Negatives: 0.15%
Neuron: Indian
   Correct: 99.61%
   True Positives: 0.00%
   True Negatives: 99.60%
   False Positives: 0.15%
   False Negatives: 0.25%
Neuron: Pacific
   Correct: 99.66%
   True Positives: 0.07%
   True Negatives: 99.59%
   False Positives: 0.00%
   False Negatives: 0.34%
```

Submission: Please zip up the entire contents of your project and submit the zip file via Canvas.

In addition to the code, you should submit a file containing the best weights that your neural network was able to learn for classifying the continents on the training data given. Don't use the test set data as training data or else you will overfit to the specific data. The grader will be using another set of test examples and your program should perform comparably on her tests to how it did on yours.