Implementation of an Artificial Neural Network Library in C

Jack Breese
TJHSST Computer Systems Lab
2007-2008

Abstract

This project aims to implement a general purpose library for neural networks in the C programming language. This library will be well-suited to basic object and pattern recognition in images, from optical character recognition to shape classification, as well as simple face recognition.

What are Neural Networks?

An artificial neural network is a computational model which emulates the biological structure of the brain. It is a system of interconnected virtual neurons which are capable of modifying their connections, adapting their responses based on accuracy. This modeling of biological networks has widespread use in the field of pattern recognition and object classification, and is well suited to tasks such as optical character recognition and junk email filtering.

Implementation

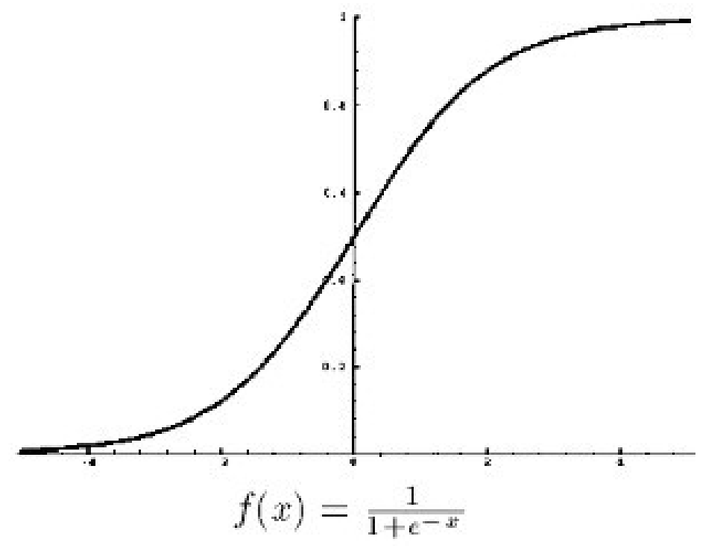
The library for neural networks is entirely implemented in C. There are several structs (shown below) which allow for a simpler way of keeping track of the network and iterating through it. Methods have been implemented for initializing networks of arbitrary size, testing said networks, performing calculations on the networks, and saving and loading the current states of the networks.

Current Running Behavior

When run, my program initializes a network of specified size, and then sets the weights on each connection to either a specified or random value. It then saves the network to a specified filename, and deallocates the memory. It then reads in the values stored in the file, and initializes a new network from that data. The initNetFromFile method then returns an array of arrays of neurons, which contains each of the layers, and the values of this newly initialized network are checked, and then saved to a different filename. The makefile which runs my program then runs diff on these two files to verify that no data has been lost in the network saving/loading process.

Sample Code

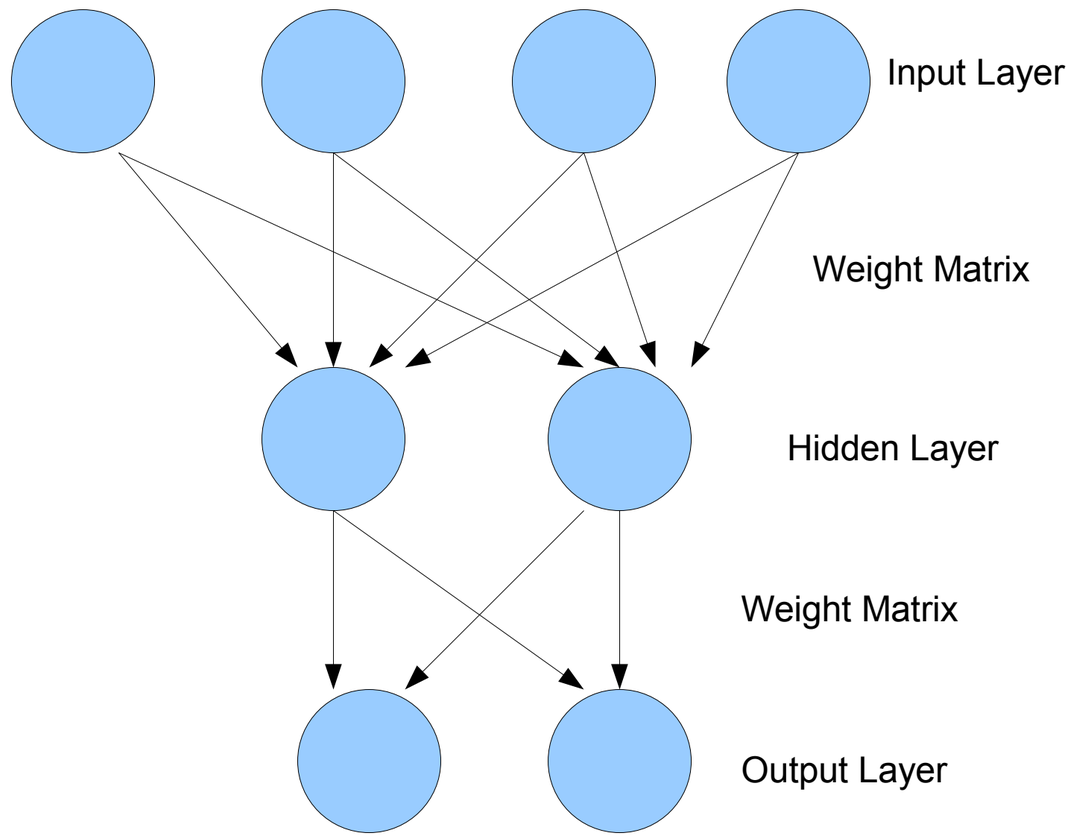
```
typedef struct _connection {
     float weight;
     struct _neuron * from;
} connection;
typedef struct _neuron {
     float d;
     connection * cons;
}neuron;
neuron* mkneuron(int c) {
     neuron* n = malloc(sizeof(neuron));
     n->d = 0;
     connection * a = malloc(c*sizeof(connection));
     n->cons = a;
     return n;
}
int saveWeights(char* filename, neuron* hidden, neuron* outputs, int insize,
int hiddensize, int outsize) {
     FILE* output;
     output = fopen(filename, "wb");
     if(output == NULL){
          fprintf(stderr, "Error: Unable to open output file for writing.");
          exit(1);
     }
     fprintf(output, "%d\n", insize);
     fprintf(output, "%d\n", hiddensize);
     fprintf(output, "%d\n", outsize);
     int i = 0;
     int j = 0;
     for(; j < hiddensize; j++) {
          for(i=0; i < insize; i++) {
               fprintf(output, "%f\n", hidden[j].cons[i].weight);
          }
     }
     i = 0;
     j = 0;
     for(; i<outsize; i++){
          for(j=0; j<hiddensize; j++){
               fprintf(output, "%f\n", outputs[i].cons[j].weight);
          }
     }
     fprintf(output, "%s\n", "[End]");
     fclose(output);
     return 0;
}
```



$$f(x) = \frac{1}{1+e^{-x}}$$

1.1 The Sigmoid Function

1.2 An Example Two-Layer Perceptron Neural Network



Sample Program Run

```
./fr
Debug: Successfully allocated memory for neural
network.
Value of weights of the hidden layer in Network 1:
  0.450000
Saving neural network, please wait...
Succesfully saved neural network.
Succesfully deallocated memory for network.
Initializing network from file...
Successfully read in file and initialized empty
network.
Value of weights of the hidden layer in Network
initialized from file after network deallocation:
 0.450000
Saving neural network, please wait...
Succesfully saved neural network.
Run diff testfile.test testfile2.test to check for
differences between the first and reinitialized
networks.
diff testfile.test testfile2.test
```