Final Project Proposal
A Simple Artificial Neural Network Written in C++

Liam Mulhall
Instructor: Shayon Gupta
TA: Alexander Curtiss
Data Structures

2018-12-10

1 General Idea

For my final project, I intend to create a simple neural network in C++. I don't mind if my program isn't terribly impressive, but I do want it to be correct and efficient. My main goal is to understand the basics of neural networks.

2 Design

The design of the neural network is fairly simple. We have a layer of input neurons, a layer of hidden neurons, and a layer of output neurons. Any given neuron has a weighted connection to every neuron in the next layer.

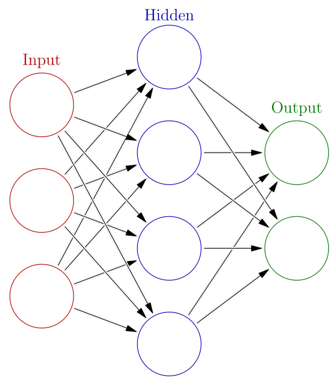

Figure 1: A Simple 3-4-2 Neural Network

The input neurons will simply hold input values. The hidden neurons will receive inputs, perform mathematical operations on those inputs, and then they may send the modified inputs to neurons in the next layer, which is the output layer in our case.

If we want the neural network to train itself, we have to use something called backpropogation. In backpropogation, we compare the received output to the desired output, and then we adjust the weight of the connections, which are used in the mathematical operations that ultimately determine the output.

Each layer except for the output layer will have a bias neuron. Bias neurons always send an output (typically 1) to the neurons in the next layer.

3 Underlying Data Structure

The underlying data structure of our neural network will be an implicit weighted graph. It will be weighted because each connection has a weight associated with it.

4 Problem

The problem we hope to solve is the XOR problem. We hope to train our neural network to behave like an XOR gate.

INPUT		**OUTPUT**
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Table 1: XOR Truth Table

5 Features

The "Network" class will have the following functions:

1. A constructor.
2. `networkFeedForward` — A function that is used to generate outputs.
3. `backPropogate` — A function that compares desired output with received output and then adjusts weights accordingly.
4. A variety of functions that print various results and statistics.

The "Neuron" class will have the following functions:

1. A constructor.
2. Getter and setter functions for the output value.
3. `neuronFeedForward` — A function that is used to generate outputs.
4. A variety of mathematical functions for manipulating inputs.

6 Self-Imposed Deadlines

This program can be broken up into three parts: (1) the “Network” class (2) the “Neuron” class, and (3) working out the details of training the network, i.e., making a training file, reading from said file, and making the output of the program look nice.

Since I’ll have time to work on this over fall break, I’ll try to make the “Optimistic Deadlines.”

Optimistic Deadlines

1. 2018-11-20
2. 2018-11-24
3. 2018-11-26

Realistic Deadlines

1. 2018-11-26
2. 2018-12-02
3. 2018-11-08