CS 480: Assignment 05 Report
Zachary Ferguson

Neural Networks

Design and Implementation:

The required implementations for this assignment closely resemble the formulas and pseudo-code provided in the lecture notes. The provided linear algebra functions eased in creating and propagating the neural network. Utilizing the formulas, and the convenient let*, I was able to create the functions in a sequential easy to read manor.

Noticing that I first need to forward propagate the data in order to back propagate the data, I created a new helper function. This helper function, forward-propagate-helper, operates the same as forward-propagate, but returns the a list of the form (hidden, output). This helps simplifies my code and reduce the chance of error.

An import step I overlooked while first implementing the network was the conversion from the provided data to the expected format. The most important step was to shuffle up the data. This was extremely import for the wine data that was sorted by the classification. This meant the generalization was horrible because it was not training on a good training set.

Reflection:

This project helped to reinforce the algorithms behind neural networks, and the formulation involved in forward and backward propagation. The biggest barrier for starting this project was understanding what the parameter where and how the equations manipulate this data. Once, I got past that implementing the functions was as simple as translating the psuedo-code.

Analyzing the Results:

Building a neural network over the provided NAND and XOR examples, I was able to test my functions. Creating a good neural network, however, requires multiple runs because the network tends to get stuck in local optimum. With some proper parameter tweaking, I was able to properly model the NAND and XOR within an acceptable error of ~10^{-4}. Through testing, I continually ran into floating point overflow issues because I had failed to scale the data. This problem was fixed and the program is now able to generalize over the provided examples.

Using the simple generalize function I was able to test the accuracy of the neural network. For the voting records, I obtain a average error of ~0.02 when forward propagating the second half of the input data. For the mpg example, I obtain an average error of ~0.002, and for the wine example, I obtain an average error of ~0.02. For the first two examples the classification are binary, 0.1 or 0.9, so the error for getting the wrong answer is equal to

$$\frac{1}{2}(0.9-0.1)^2=\frac{1}{2}(0.64)=0.32$$

Therefore, if 100% of the classifications where incorrect, the average error 0.32. Using this knowledge, for the voting records ~0.625% were incorrect, and for the mpg example ~3.125% were incorrectly labeled. This percent error is acceptable and will varies slightly based on the parameters.

For the wine example, however, the output is one of three classes encoded as corners of a 3D hypercube. That is the set of all classes is equal to {(0.9, 0.1, 0.1), (0.1, 0.9, 0.1), (0.1, 0.1, 0.9)}. So the error for getting the wrong classification is equal to

$$\frac{1}{2}\left(\left[0.8\ 0.0\ -0.8\right]\begin{bmatrix}0.8\\0.0\\-0.8\end{bmatrix}\right)=\frac{1}{2}\left(0.64+0.0+0.64\right)=0.64$$

Therefore, if all classification are wrong the average error will be 0.64. Using this knowledge, we can compute the percent classifications wrong for out neural network. For the wine example ~4.69% of the testing set was incorrectly identified.