Colin Hwang

Neural Network Program Write up

**Tic-Tac-Toe Endgame Data Set**

The interesting data set that I used in addition to the data sets provided was a tic-tac-toe endgame data set provided from [UCI’s machine learning repository](https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame). The dataset provides a complete set of possible tic-tac-toe board configurations at the end of tic-tac-toe games. Typically, tic-tac-toe is played with x’s and o’s. This dataset aims to find out endgame scenarios in which the player playing “x” piece wins. An important constraint to this data set is that the player using x’s moves first. With this constraint, there are a total of 958 possible end game board configurations. Out of these 958 possible end game boards, 626, or 65.3%, of the endgame board configurations result in wins for ‘x’. 332, or 34.7%, of the end game board configurations result in wins for ‘o’.

The data set provides 9 attributes, each corresponding to a different square in the tic-tac-toe board. Each attribute can be filled with three different values: a ‘x’ to represent the player using x pieces, ‘o’ to represent the player using o pieces, and ‘b’ to represent blank squares. Unfortunately, the data set did not provide attribute values that were doubles, which makes it incompatible with the program written. Therefore, I modified the input attributes’ data by converting each string value into double value, normalizing it with respect to the number of options for each attribute. Therefore, since there were three possible options for the attribute values, the new attribute values became 0.333 for x’s, 0.666 for o’s and 0.999 for blank squares.

The data set also provides two output values. “Positive” for endgame boards in which the player using x’s wins, and “negative” for endgame boards in which the player using o’s wins. Since these also did not fall within the assignment’s specifications, I modified the output values to be 0 and 1, with 0 representing a win for player x and 1 representing a loss or draw for player x.

To create the testing and training file, I used the dataset provided, which will be attached to my submission. I decided to split the data equally between the two, randomly choosing which endgame boards to include in each of the files, but I kept the number of differing outputs, 0 and 1, close to being the same between the two files. These files will also be attached to my submission.

To create a neural network that adhered to the assignment’s specifications, I wrote a short program to write it for me. The program prompts the user for the number of input, hidden and output nodes, and then proceeds to generate the initial weights. The initial weights were generated by setting the seed of srand using the time function: srand((unsigned)(time(NULL)). Then, I used the rand function to produce pseudo randomized weights that fall between 0 and 1: (float)rand()/RAND\_MAX. This program will be attached to my submission.

The neural network parameters that I decided to include were 9 input nodes, 5 hidden nodes, and 5 output nodes. There was no real reasoning behind choosing these numbers. It was mostly random. Then, after the program generated the neural network, I trained it using my program with these parameters: a 0.5 learning rate and 100 epochs. Then, I tested the program with the testing set, and I seemed to get good results!

Text

Description automatically generated

I wanted to see what would happen if I changed the learning rate. So, I changed the learning rate to 0.1, but I got the same results. I also tried out how decreasing or increasing the number of epochs (500 and 20 epochs) would affect my results. As expected, the increasing and decreasing the number of epochs resulted in better and worse results respectively, but not by a significant amount.

**Important File Information**

Here I will list the various filenames for my dataset and a brief explanation.

**tic-tac-toeRAW.data** 🡪 Full data file with original values taken from UCI’s machine learning repository

**tictactoe.training.txt** 🡪 Use this file for training NN. Has converted values from the raw data set.

**tictactoe.testing.txt** 🡪 Use this file for testing NN. Has converted values from the raw data set.

**tictactoe.NN.txt** 🡪 Neural Network with randomly generated weights with 9 input nodes, 5 hidden nodes, 5 output nodes. Untrained.

**tictactoe.trained.txt** 🡪Neural Network trained with 100 epochs and a 0.2 learning rate.

**tictactoe.results.txt** 🡪 Results file from testing a neural network with 100 epochs and a 0.2 learning rate. Led to reasonably good results.