Machine learning final project

Members:
Harry Berman
John Hamre
Stanley Su

Introduction:

For our final project, we trained a neural network that would learn to make the best move at a crucial stage of a poker hand -- the post flop action. In poker, the flop is the point in the game where three cards are revealed to be shared by each player's hand. This is the point at which each player receives the most hand-information during a round of poker. The bot decides one of two things at this stage: either it should continue playing the hand, or it should fold the hand away if it is unlikely to win. In order for the network to learn, we generated thousands of simulation games to analyze to determine what the best move would have been in that situation. Ideally, the network would learn from these test situations so that when given a new hand, it will make the correct choice.

Experiment:

In order to train a neural network to predict the outcome from the flop, we first need a way to convert a game into data a neural net can train on. To do this, we created examples using a one-hot encoding for both the cards in the player's hand and the cards on the table. Since there are 52 distinct cards in a deck, this process results in examples with 104 features.

To generate the labels for these examples, a full hand was simulated using a modified version of this code for 7-card poker hand evaluation. If the neural net's hand (i.e. the player hand) won the simulated game, then the label would be 1.0. If instead the "opponent hand" won, the label would be -1.0. We also considered draws to be wins, since the most money would be saved in that hand by calling. Finally, we note that the Neural Net can still only "see" their own cards and the 3 cards showing on the board, the same as a player would at that point of a hand.

Since poker has a very hard pattern to learn, we decided to construct a three layer neural network and perform tests using that, as we believe that it could handle harder patterns. We adapted the three layer neural network from the two layer neural network in assignment 8.

Most of the process is similar to the 2 Layer NN except that the backpropagate function has a few adjustments. Beginning with the forward propagation initialization via the predict function, it computes outputs of neural network layers for given input examples. Subsequently, it calculates the output layer error by comparing predicted outputs to actual labels. The method then updates weights and biases for each layer in reverse order, starting from the output layer and moving towards the input layer. It modifies connections between layers, leveraging error values and activation function slopes to fine-tune weights and biases.

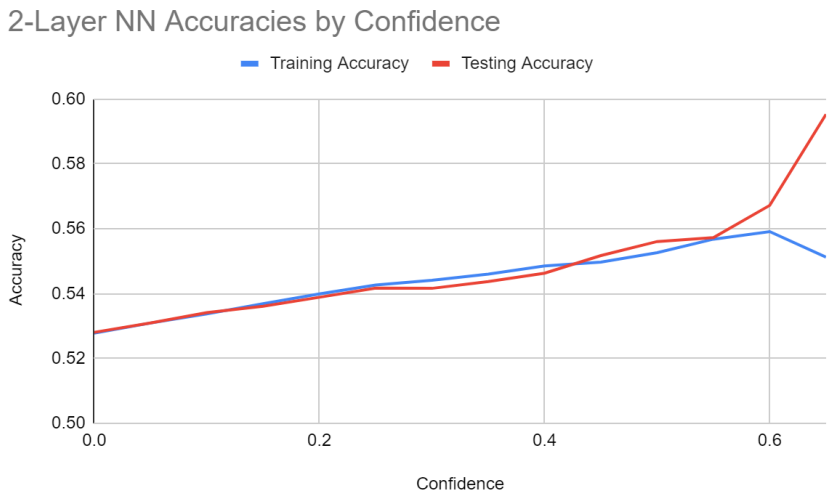
Results:

Since the training on the three layer neural network on the big dataset with 100,000 examples takes approximately 40 hours, we decided to train it on a smaller dataset with only 1,000 examples in order to tune the parameters since parameter tuning is crucial to

performance of a neural network. From the small dataset, we found that the most optimal parameters are 60 hidden nodes for the first hidden layer and 30 hidden nodes in the second hidden layer, 350 iterations and a learning rate of 0.0275. 10-fold cross validation was used on our generated dataset for this experiment. Without a confidence threshold, we achieved both a training accuracy of approximately 53.2% and a testing accuracy of 56.4%.

For comparison, we also trained a 2-layer neural network on a similar dataset of 100,000 randomly generated sample games. After some basic parameter tests, we determined that 20 hidden nodes, 25 iterations, and a learning rate of 0.05 were best for this problem. 10-fold cross validation was used on our generated dataset for this experiment. Without any confidence threshold, we achieved both a training and testing accuracy of approximately 52.8%.

For additional statistics, we fixed a confidence threshold In the chart below, the results of excluding examples from the accuracy calculations below a certain confidence threshold are shown below. No examples tested had confidence levels higher than 0.7, so these values are excluded from this chart. Excluding the value at 0.65, which was based on an extremely small sample size, the best results achieved were at a confidence threshold of 0.6, with a 56.7% accuracy.



Conclusion:

Through our testing, we found that the three layer neural network outperformed the two layer network by about 3.5 percent. This helps corroborate our hypothesis that a three layer neural network would be more equipped to deal with the problem we are trying to learn for. Also note that the three layer neural network trained on a significantly smaller data set: 1,000 examples versus 100,000 examples. In the future, we plan on training the neural network with the larger data set and editing the hyperparameters in order to optimize the network as best we can.