# Neural Network Training on Poker Hand Classification Data

We chose to train a neural network to classify the data because the data has ten features. These features themselves have a large range (1-13). Neural networks are very good at adapting to extremely varied data.

There are two major attributes to each card in the five cards that compose a hand. There’s the rank and the suit of the card. We found that the suit only matter in 3 out of the 10 classifications of a hand. This led us to put more preference on the rank of the cards.

The training data was split into a rank array and a suit array. Two neural networks were trained with one using the rank data and the other using the suit data.

Since 50.117739% of the results sat in the first class and 42.256903% sat in the second class we tried to level the data to account for the other 8.6%. This later proved ineffective because it put too much weight on the 8.6% that rarely showed up in the set.

We chose a deep network for the rank neural network because of the 7 different classes it had to classify. The shape that best suited it was [5,15,10,15,8,1]. After a couple of tests the best learning rate for the rank neural network was 1.2.

This was the confusion matrix after training:

[5., 0., 0., 0., 0., 0., 0., 0., 0., 0.]

[0., 5., 0., 0., 0., 0., 0., 0., 0., 0.]

[0., 0., 5., 0., 0., 0., 0., 0., 0., 0.]

[0., 0., 0., 5., 0., 0., 0., 0., 0., 0.]

[0., 0., 0., 0., 5., 0., 0., 0., 0., 0.]

[0., 0., 0., 0., 0., 0., 0., 3., 2., 0.]

[0., 0., 0., 0., 0., 0., 5., 0., 0., 0.]

[0., 0., 0., 0., 0., 0., 0., 5., 0., 0.]

[0., 0., 0., 0., 0., 0., 0., 1., 4., 0.]

[0., 0., 0., 0., 0., 0., 0., 2., 3., 0.]

Correct Percentage: 78.0

Wrong Percentage: 22.0

We used the shape [5,8,6,4,1] for the neural network of the suits. The learning rate was 1. This proved very effective as we were able to consistently train a neural network that could accurately classify the suit of each hand.

[5., 0., 0., 0., 0., 0., 0., 0., 0., 0.]

[0., 5., 0., 0., 0., 0., 0., 0., 0., 0.]

[0., 0., 5., 0., 0., 0., 0., 0., 0., 0.]

[0., 0., 0., 5., 0., 0., 0., 0., 0., 0.]

[0., 0., 0., 0., 5., 0., 0., 0., 0., 0.]

[0., 0., 0., 0., 0., 5., 0., 0., 0., 0.]

[0., 0., 0., 0., 0., 0., 5., 0., 0., 0.]

[0., 0., 0., 0., 0., 0., 0., 5., 0., 0.]

[0., 0., 0., 0., 0., 0., 0., 0., 5., 0.]

[0., 0., 0., 0., 0., 0., 0., 0., 0., 5.]

Correct Percentage: 100.0

Wrong Percentage: 0.0

The rank neural network had an average accuracy of 77% while the suit neural network had an average accuracy of 98%.

These neural networks then provided input into a third neural network that processed the data and reported a final guess. The shape for the final neural network was [2,16,8,4,1]. With the learning rate set to 1, the neural network had an average accuracy of 10%. It was polarized to account for the 8.6% of the varied data.

The best result we got was training a neural network with the full training set. The neural network’s learning rate was set to 1 and the network shape was [10, 8, 6, 4, 1]. We trained the network with varying training iterations but they did not make any difference in the accuracy of the network.

The confusion matrix for this network was:

[24905., 0., 0., 0., 0., 0., 0., 0., 0.]

[21255., 0., 0., 0., 0., 0., 0., 0., 0.]

[2406., 0., 0., 0., 0., 0., 0., 0., 0.]

[1040., 0., 0., 0., 0., 0., 0., 0., 0.]

[218., 0., 0., 0., 0., 0., 0., 0., 0.]

[104., 0., 0., 0., 0., 0., 0., 0., 0.]

[62., 0., 0., 0., 0., 0., 0., 0., 0.]

[9., 0., 0., 0., 0., 0., 0., 0., 0.]

[1., 0., 0., 0., 0., 0., 0., 0., 0.]

Correct Percentage: 49.81

Wrong Percentage: 50.19

Since most of the results are situated in the first and second class, the network was skewed to the first class. Guessing 0 for all the tests gave an accuracy of 49.81%. This approached was smarter than the other optimization strategies that we tried.