In this version I opted to use a neural network to train my poker agent. My multilayer perceptron neural network took in 4 input values and put out 3 output values. I used a Sigmoid transfer function and a backpropagation learning rule. The input values that I chose to use were normalized values of the first pocket card and the second pocket card, a binary indicator if the card suits match, and a normalized position value. I chose to represent each card in values within the range of [0..1]. To do this I took the mod of the integer value of the card and 13, and then divided the result by 13. This enabled me to maintain rank value (Ac= 12=0.923 vs 3c=0.077) while also making it easier for the network to interpret. For the normalized position value I took the current seat value and divided it by the number of remaining players. As for results, I created 3 different output neurons, one that fired for a fold, one for a call or a check, and another for a raise or a bet. Only one neuron will contain a firing value of 1 while the others will maintain their values of 0. This creates the typical XOR ANN. Using a 0.3 learning rate and 0.2 max error, I let the training session run all night and after 861,356 iterations obtained an error rate of 0.1537.