Deducing Relevant Bridge Bidding Information from Double Dummy Data

Daniel Winograd-Cort

Abstract

The game of Bridge has entertained card players for many years. The players start by evaluating their hands individually and then using this information to bid for a contract. Bidding systems provide a way for partners to communicate the values of their hands so they can find an appropriate contract, but they are often imprecise and can lead teams into contracts that they cannot keep. Unfortunately, creating a new bidding system is a challenging task. There are approximately 10^{28} ways to deal 13 cards each to 4 people from a 52 card deck, but there are 10^{47} possible bidding sequences (Butler, 2008). Therefore, instead of creating a new bidding system, I propose to determine what information is most important to convey in order to achieve success. As opposed to hand evaluation, this system will use relationships between two partners' hands. For this project, I train neural networks using sample simplified hands of Bridge (known as Double Dummy Bridge results) to find which particular patterns and relationships of cards represent the necessary information needed to complete the best possible contract. Although I am unable to extract meaningful rules from the neural networks to find these combinations, I show the proper steps and the places for improvement.

Introduction

The game of Bridge is not a perfect information game, and as such, has no chance of being perfectly solved. Furthermore, there is no proof that current strategies are optimal, so it may be possible to develop new strategies and systems superior to those currently in use. In this paper, I will explore possible enhancements to hand evaluation and bidding. I will assume a passing knowledge of Bridge play and terms for this report; if the reader is interested in more information, please see the supplementary Appendix of Bridge Terminology or the plethora of online and in print Bridge guides.

Bridge Information

Almost all current Bridge bidding systems rely on a player evaluating his hand, determining the hand's value (typically some number), and making a bid that communicates this value to his partner. The evaluation must be complex enough to be meaningful, but simple enough to both easily calculate and communicate during bidding. This must be done under the restrictions of the game of Bridge; the bids are allowed to have special meanings, but there is only a small, finite set of bids that each player may make.

The most popular hand evaluation system is known as the High Card Point (HCP) system. In the HCP system, a hand's value is the sum of its cards where each ace is worth four points, each king is worth three points, each queen, two points, each jack, one point, and all other cards are worthless. If a team has a combined score of over 26 points, then they should attempt to bid to a game contract (Rubens, 1969).

An experienced Bridge player will recognize that this is an oversimplification of the HCP system. To fine-tune your hand score, it is necessary to account for the distribution of suits in your hand, whether your strongest suit is the same as your partner's or perhaps one of your opponent's, and many other small factors. In addition, if there is going to be a trump suit, the players want to assure that they have at least eight of the thirteen trump between them.

Are we missing something? Maybe extra points should be awarded if I have a long suit where my partner has none of that suit. Perhaps points should be subtracted if I have a king and my partner has neither the ace nor the queen (this is a scenario that could lead to a finesse, a play where the king is denied taking any tricks by the opposing team). Unfortunately, since HCP and other hand evaluation systems look only at each individual hand, they won't account for any of these things.

I will define an information system to be a set of rules based on relationships in cards between two teammates' hands that, when satisfied, predicts success accurately. If my partner and I each have few cards in a suit, HCP tells us to add extra points. However, more points should be awarded if those suits are different than if they are the same. HCP does not account for which suit is which as it does not compare between hands; a good information system, on the other hand, would account for such scenarios.

Experimental Technique

The goal of this project is to determine an information system that can be used by a new bidding system to allow better bidding accuracy. A decision tree seems like the obvious choice at first, but this proves problematic. In order to discover a set of constraints like this, one would need to branch on every detail about the hands. For example, one way to branch would be to ask at each node, "Does the hand contain this card?" for every possible card. If I branch on the cards of just one hand, the tree would have 2^{52} outputs, and unfortunately, generating

enough sample data to extract useful information from this tree would take on the order of millennia. A solution to this problem may be to choose the set of information one thinks is correct and use the tree to check how important it is. However, it would be difficult to determine which set to use. Decision trees are appropriate for verification purposes, and in fact, decision tree based algorithms are used to test new strategies, but they are not useful for generating new strategies.

A neural network, however, can be trained using all of the cards in the hands as its input without needing too many training samples. As it learns, it will identify the importance of various features of the hand; each hidden node in the network will identify a relationship about the inputs and affect the output proportionally to how important that relationship is. Thus, I will attempt to create a neural network that can identify when its input hands will lead to success and then extract the most important rules from it.

Experimental Design

General Neural Network Design

Neural networks are fickle constructs and have many adjustable parameters that require fine-tuning to maximize the network's efficiency. I have designed seven networks with common trends, which I describe below.

Structure

Each network is composed of a layer of input nodes, a single layer of hidden nodes, and a single output node that indicates success or failure. Neural networks have the capacity to be much more complicated than this, and if my goal were to determine, given two input hands, the chance of success, I would use a more complex neural network. However, my goal is to extract simple and meaningful rules from the network, and as the network becomes more complicated, the rules become more obfuscated.

The networks I test will vary in how many hidden nodes they have and how they are connected to the inputs. Although one of the primary reasons for choosing neural networks for this project is so that I would not have to exert any influence on how they learn, in practice, some guidance is necessary. Meaningful rules may prove too difficult to extract from fully connected neural networks, even with only one hidden layer.

Training

Because Bridge is not a perfect information game, a minimax algorithm cannot be used. Moreover, it is not even certain that every player will make optimal plays. Therefore, for the purposes of this project, I will be analyzing perfectly played hands of Double Dummy (DD)

Bridge rather than actual Bridge. DD Bridge is a perfect information variant of Bridge that can easily be solved for each hand. It is so similar to actual Bridge that it is commonly used to predict "par" for the hands.

Although DD Bridge is easy to solve, it is still time consuming. Luckily, Dr. Matt Ginsberg, creator of the GIB Bridge AI (Ginsberg, 2001), has graciously provided me with a database of over 700,000 deals and their DD results. Thus, all of the networks will be trained using supervised learning with results from these solved DD hands.

The learning rates and momentums will be adjusted as necessary as a fine-tuning step of network creation and training, and an annealing algorithm is used to help find good minimums as well. The number of training patterns will also be varied to maximize the accuracy of the trained system: too few training examples could result in inaccuracies due to insufficient data, but too many examples could cause problems with overtraining. Only data from the best-trained networks is presented.

Performance Analysis

There are two types of errors that any binary system can produce: it can predict success when it should predict failure, and it can predict failure when it should predict success. I will refer to these as "set errors" and "miss errors" respectively because a team will get "set" if the system inaccurately predicts success and will "miss" an opportunity if the system inaccurately predicts failure. In the game of Bridge, these errors should not necessarily be treated equally. Missing the potential of a hand because of an overly strict system is bad, but bidding too high and getting set is often much worse. Luckily, the output of a network is a real number between zero and one where zero indicates failure and one indicates success. When analyzing this data, I need to choose an appropriate cutoff to minimize the total error. Therefore, each system tested will output as its result a graph of its errors as a function of this cutoff.

To measure the performance of a given network, I will compare it to that of a modified HCP system (the team needs at least eight trump as well as 26 high card points to complete their game contract).

Rule Extraction

If a network performs exceptionally well, I will apply the rule extraction techniques on it. I have three methods for extracting meaning from a trained neural network:

Zeroing Weights

I assume that the important information that a node receives has a large magnitude weight. Therefore, to simplify the node, I will set small values to zero. This can diminish the accuracy, but it is a fast way to simplify a set of hidden nodes.

Rounding Weights

Similar to zeroing weights, the process of rounding weights allows all weights to be only a certain set of values. This does not have as significant an impact on the performance as zeroing the weights as it leaves more information that the network has learned intact. However, the more information the network has, the harder it is to find patterns in it that will lead to rules.

Examining Hidden Values

I will examine individual hidden nodes to deduce their effect on the final output of the network. One way to determine influence on the final output is by examining the ratios of weights of the inputs to the final output: a large weight indicates a strong influence. Also, if a node mimics the value of the final output closely, it may be of primary importance where the others are there mainly to correct some of its mistakes. Alternatively, if a node has a very small weight or comes out consistently close to zero or one, then it does not have a significant impact on the final result.

Implementation

All code for this project is written in Java. I utilize two outside libraries to assist in the coding: Joone (Marrone, 2008) and Jama (JAMA: Java Matrix Package, 2005). Joone, a Java Object Oriented Neural Engine, allows me to create neural networks of varying structure quickly and easily. I simply position the nodes and connections, supply training information and learning parameters, and run the neural network for the desired number of generations. It uses a form of gradient descent to lower the root mean square error of the outputs. Unfortunately, Joone was designed to make neural networks that solve problems rather than neural networks that are used to find rules. Ultimately, I use Joone to train the network and then Jama, a Java matrix package, to examine the weights and verify their accuracy.

Input Data

To train the network, I want to use information about the partners' hands as input and whether they will complete their contract as output. The hands themselves will dictate which contract the partners will try to complete, the nature of the bidding system will direct them to an ultimate bid, and their opponents' hands will impact how many tricks they take. In fact, even if we fix two players' hands and choose an appropriate contract, there is no sure way to tell if they will succeed because the opponents hands are still unknown. To avoid these complications, we will be using Double Dummy results instead.

To further simplify the data, we will consider only the information necessary for a team to take ten tricks when Spades are trump (this is known as a game bid in a major suit and has significance in scoring).

I formatted the data from Matt Ginsberg's Double Dummy database into 104 binary digits representing whether, for each card, the player or his partner had it. That is, the first 52 bits represent each card and if the first player has it, and the second 52 bits represent each card and if his partner has it. Unfortunately, there are redundancies in this data (clearly, if one player has a card, the other certainly will not), but neural networks train much better on binary systems than trinary ones. Success is a binary digit based solely on whether the partnership can make 10 tricks with Spades as trump.

Tested Neural Networks

I have chosen seven neural network designs that may provide good results when trained and examined. I will briefly describe them here.

Full 50 Network

The 104 input nodes are fully connected to 50 hidden nodes that all feed directly into the output node.

Full 5 Network

This network is similar to the previous one, but here only five hidden nodes to simplify rule extraction.

High Network

This network only connects the high cards (10, jack, queen, king, and ace) and the trump suit cards to the five hidden nodes. It is similar to the Full 50 and Full 5 networks, but all connections from a low card in a non-trump suit are zero, and so those inputs are ignored. The purpose of this net is to find rules specifically having to do with trump length and high cards. Hopefully, the result will be similar to, if not better than, the HCP model.

Custom 34 Network

This network has many specified hidden nodes that may take on special tasks. The first and second hidden nodes receive information only from the trump cards for the first and second player respectively. The third and fourth hidden nodes receive information only from the jacks, queens, kings, and aces of the first and second player respectively. The next 10 nodes have access to all of the first hand, and the following 10 nodes have access to only the second hand. The final 10 nodes are fully connected.

Custom 9 Network

This is identical to the previous except that it has only 2 nodes for each hand and 1 node for all of the inputs.

Distribution Network

This network examines only the distributions of the suits in the hands. The data is pre-processed so that there are eight sets of inputs, each corresponding to a suit in one of the hands. Each set of inputs consists of 14 values where the first represents having 0 cards of the given suit in the hand, the second represents having 1 of that suit, and so on. Thus, for each set of 14 values, one will be set to one and the others to zero. Each suit's 14 inputs is connected to a different hidden node, and, as usual, all of the hidden nodes are connected to the output.

HCP Seeded Network

This network is designed to model how HCP works. It has two hidden nodes, one for high card points and one for trump cards. Further, its weights and biases are pre-seeded to match the HCP. That is, it counts each trump card as 1 and returns success only if there are more than 8 total. Also, it weights the high cards just as HCP does, and it returns success only if there are more than 26 points.

Results

An Analytical Look at HCP

Using a simple decision tree on the Double Dummy results, I was able to determine the set error rate and miss error rate with various HCP demands. These graphs are shown below in figures 1 through 3.

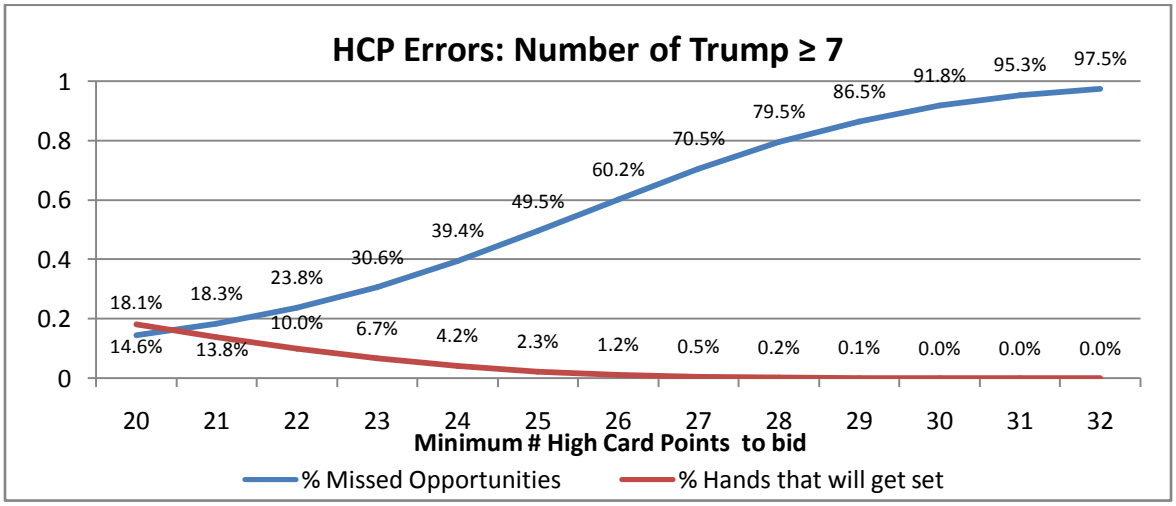

Figure 1

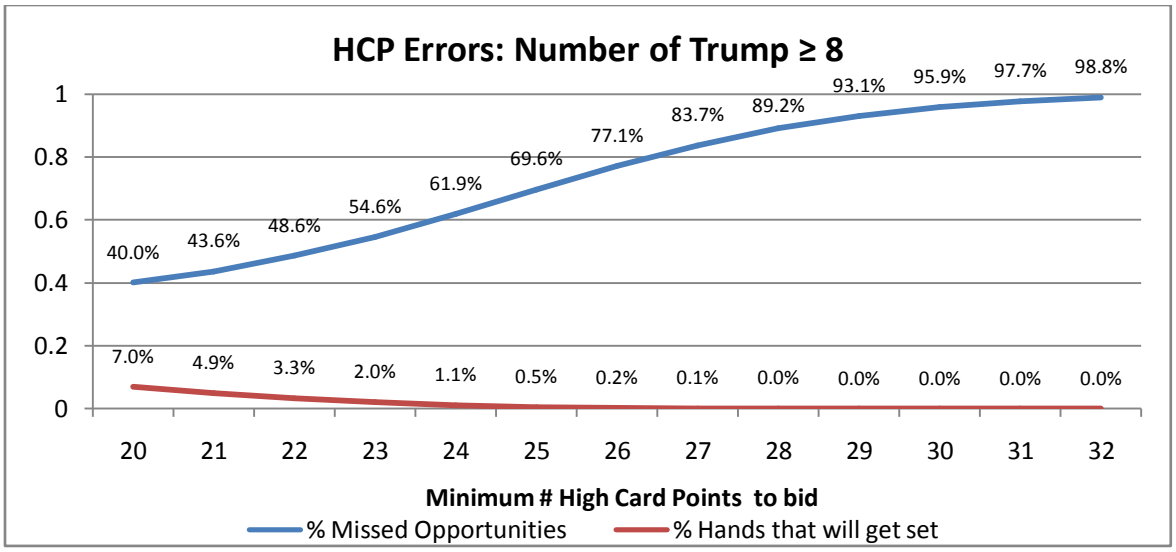

Figure 2

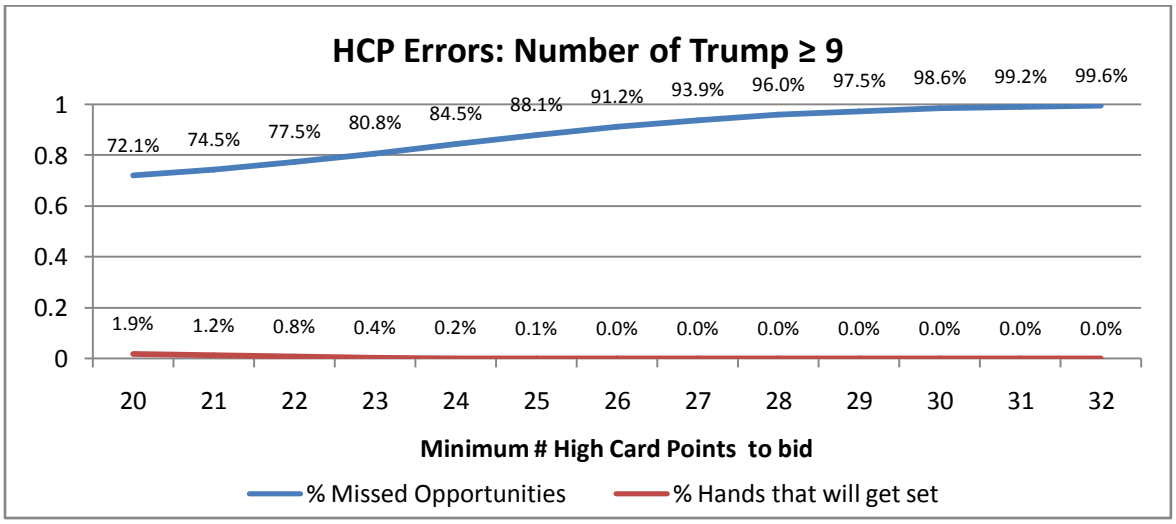

Figure 3

Performance

The graphs below (Figures 4 through 10) show the set error rate versus the miss error rate of the seven neural networks:

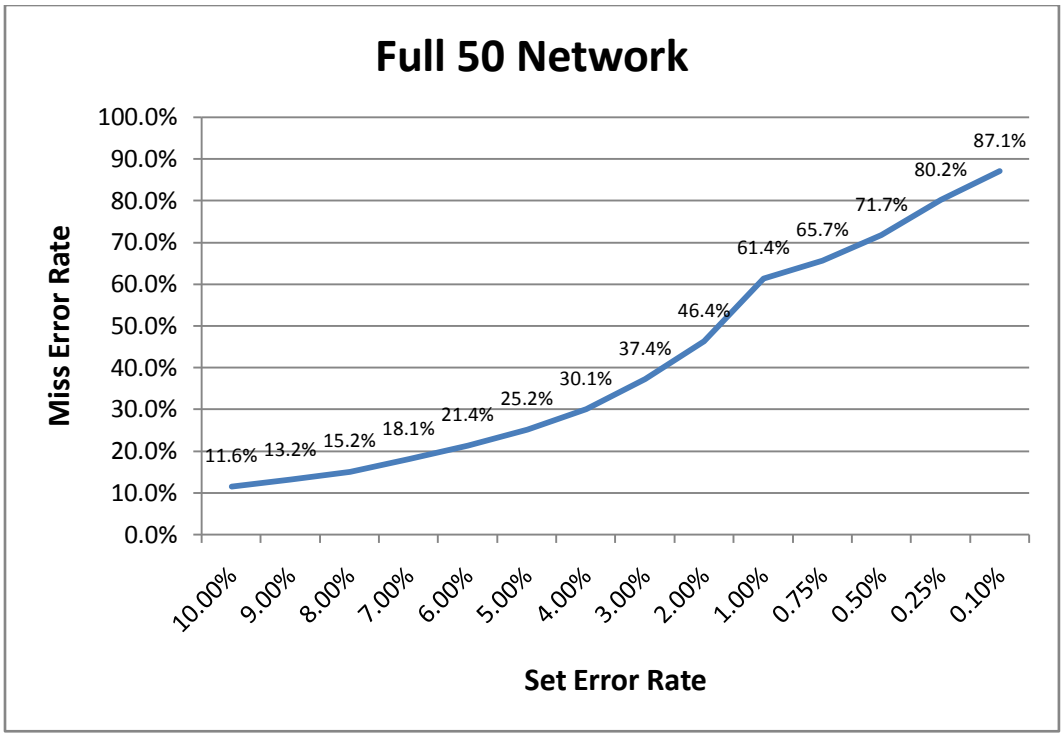

Figure 4

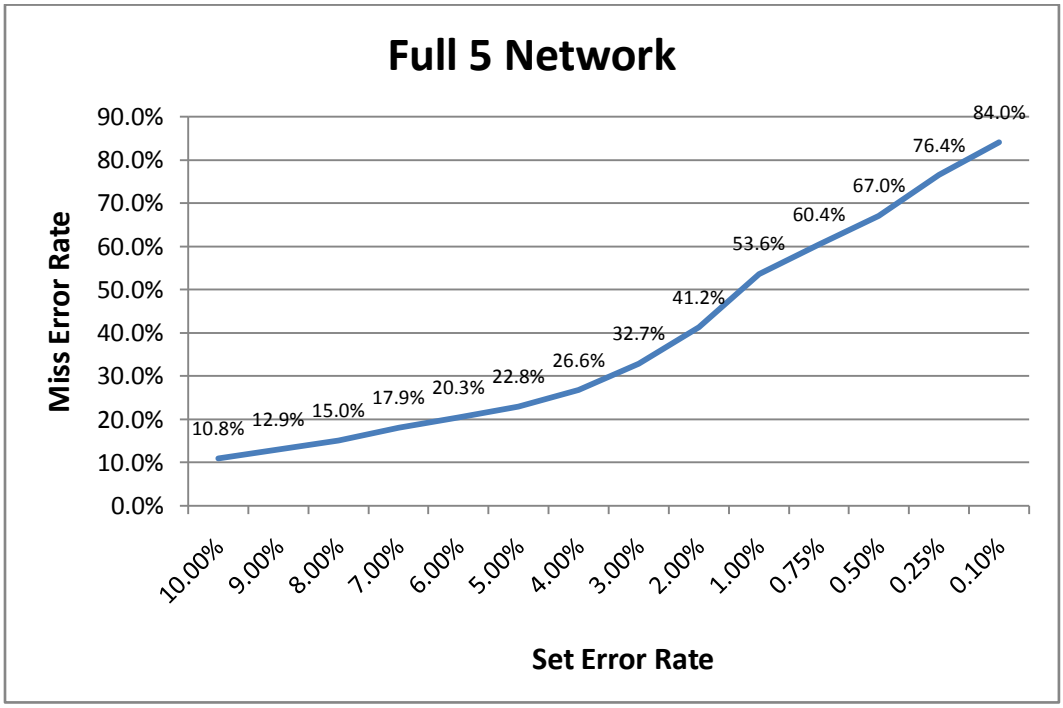

Figure 5

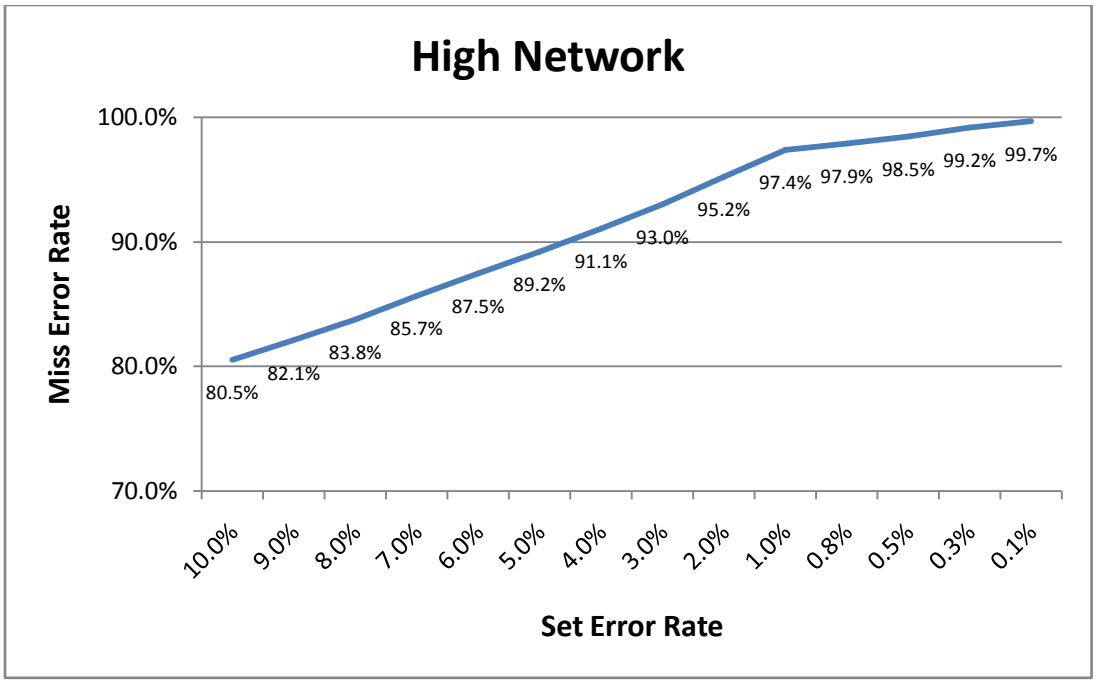

Figure 6

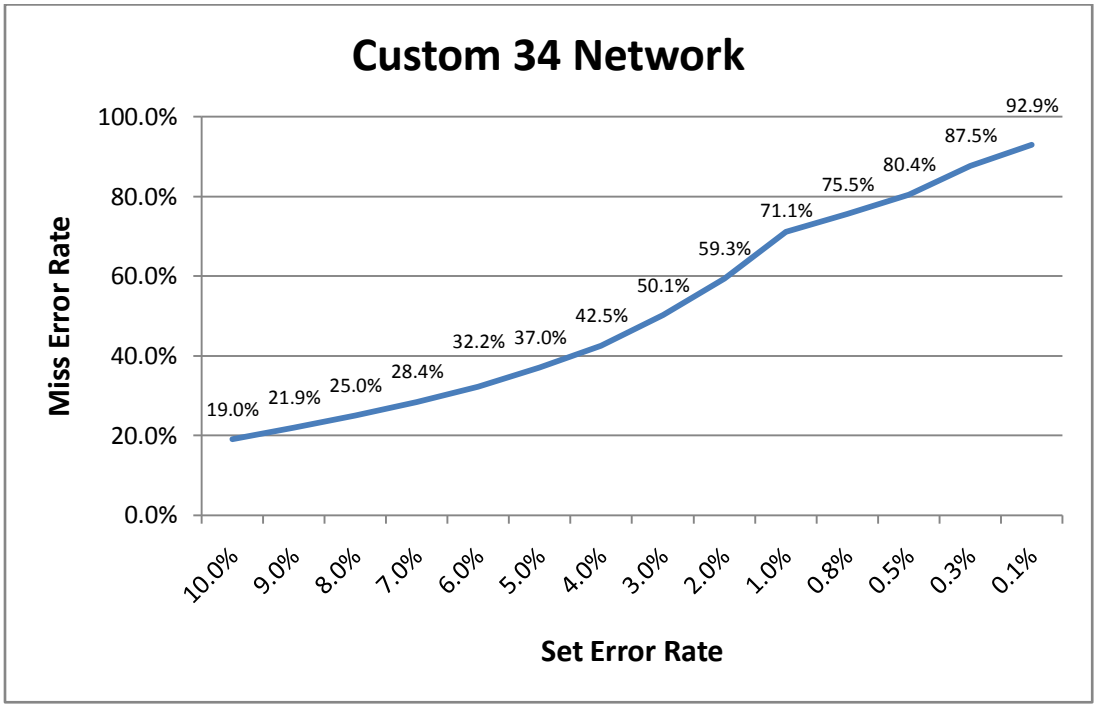

Figure 7

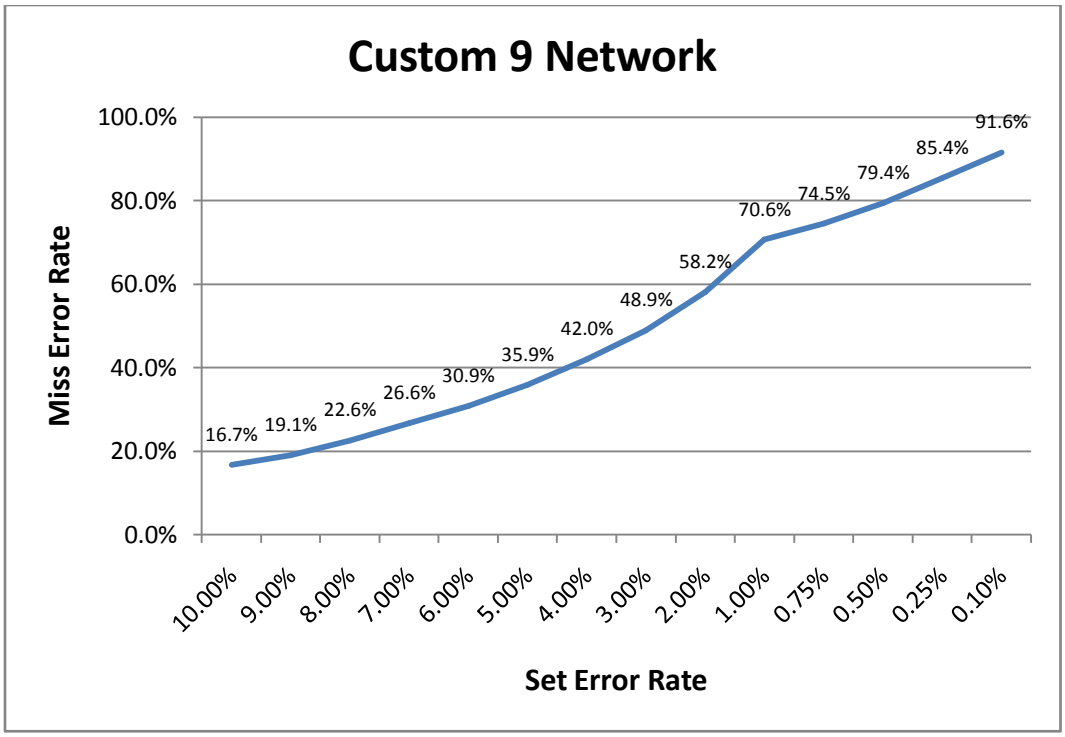

Figure 8

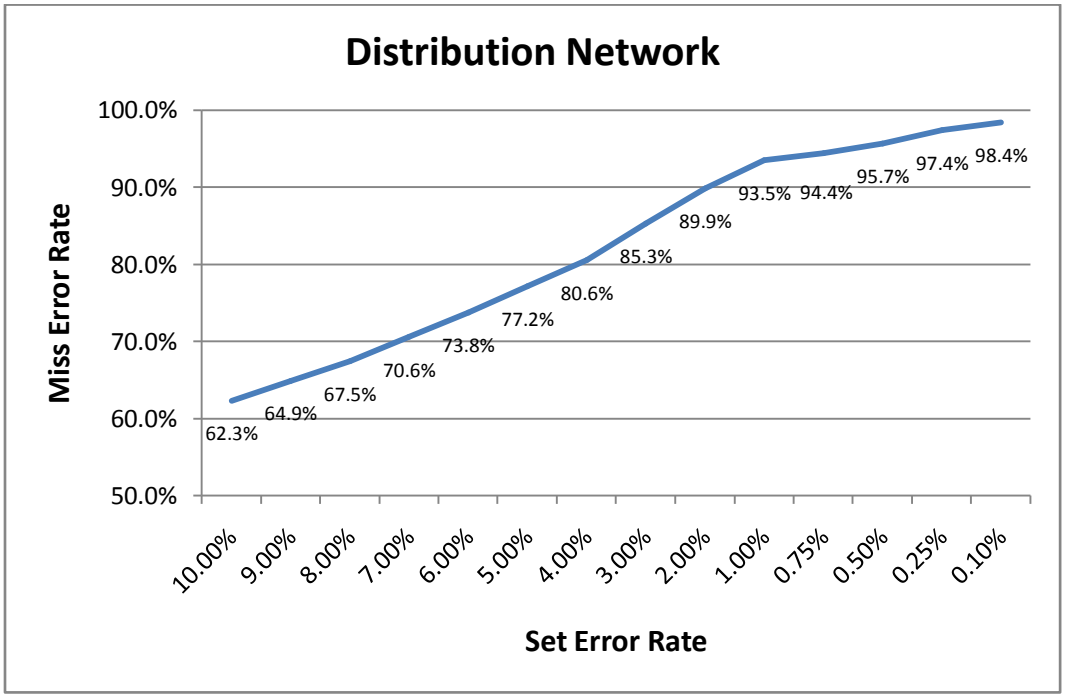

Figure 9

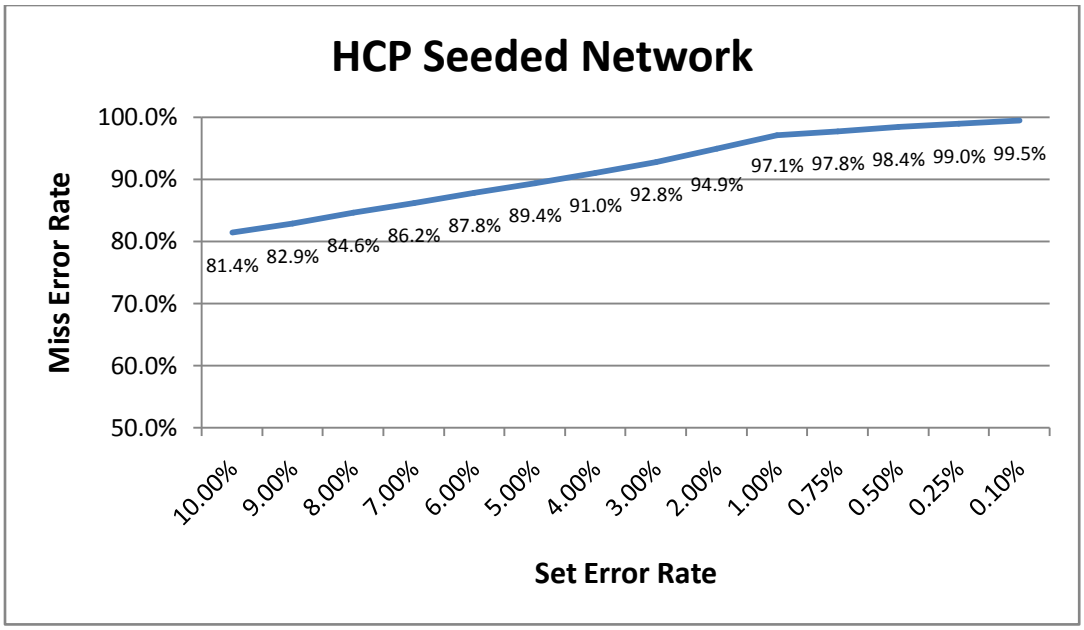

Figure 10

Rule Extraction

The two most promising networks for rule extraction are the Full 5 Network and the Custom 9 Network. Rule extraction results follow here.

Full 5 Network

The following graph (Figure 11) shows the network with all weights with magnitude less than 15 set to zero:

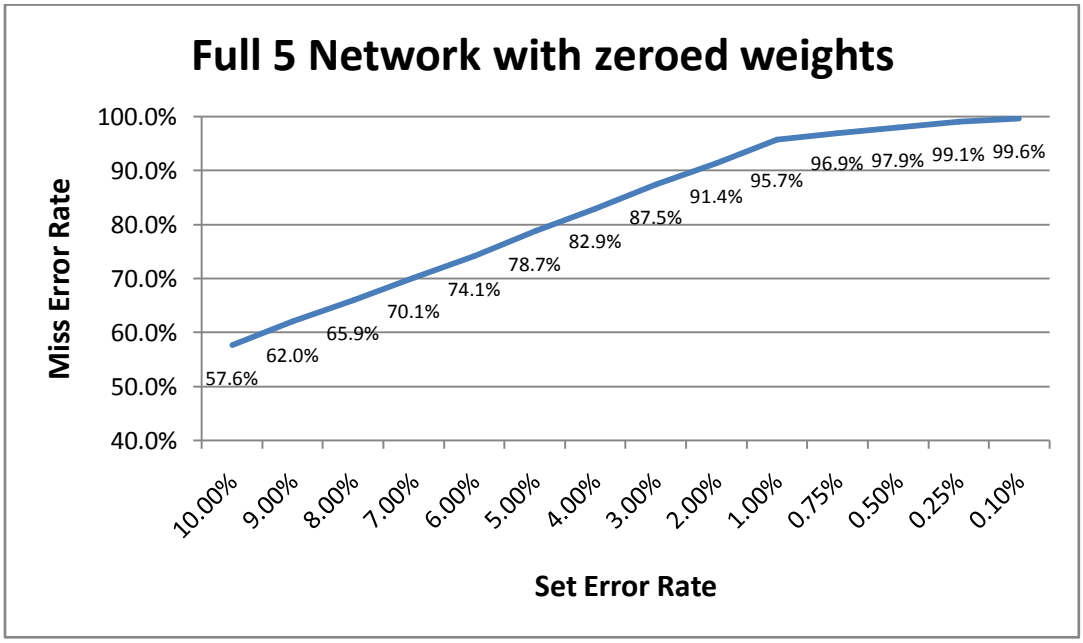

Figure 11 Full 5 Network with all weights ≤ 15 set to zero

This next graph (Figure 12) shows the network with the weights rounded to the nearest 10 (i.e. -40, -30, …, 30, 40):

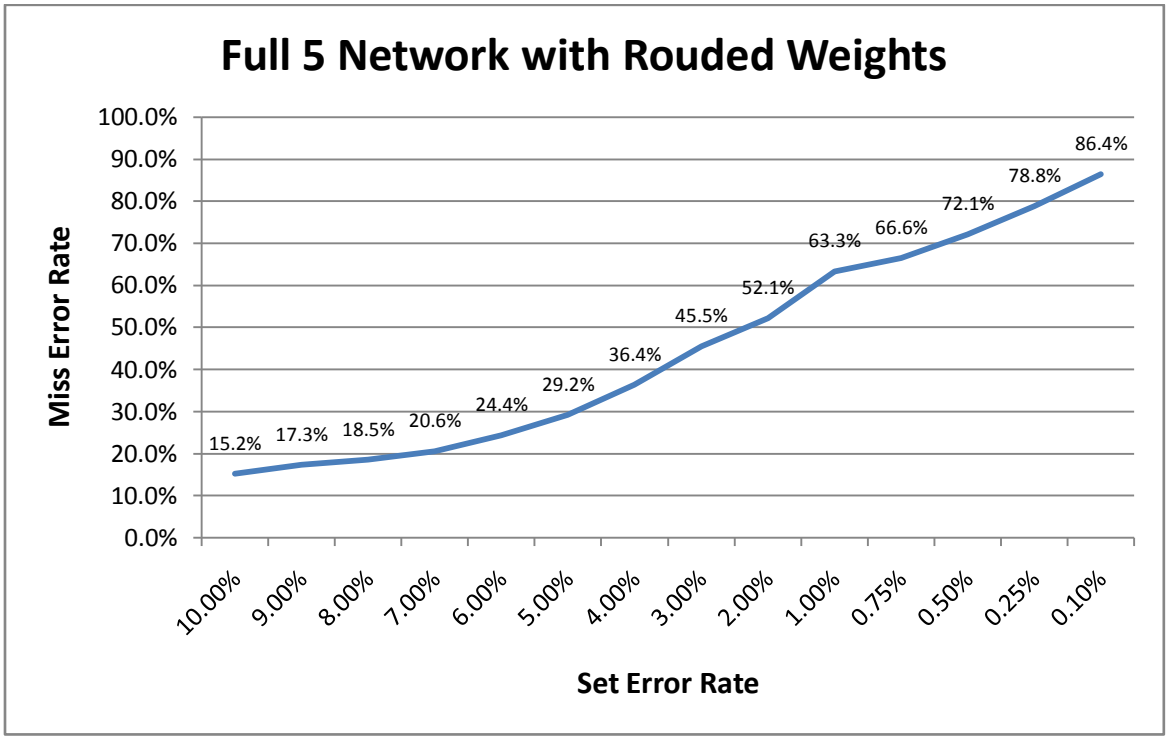

Figure 12 Full 5 Network with all weights rounded to $\{-40,-30,-20,-10,0,10,20,30,40\}$

This final rule extraction graph (Figure 13) shows the individual hidden nodes and their error rates:

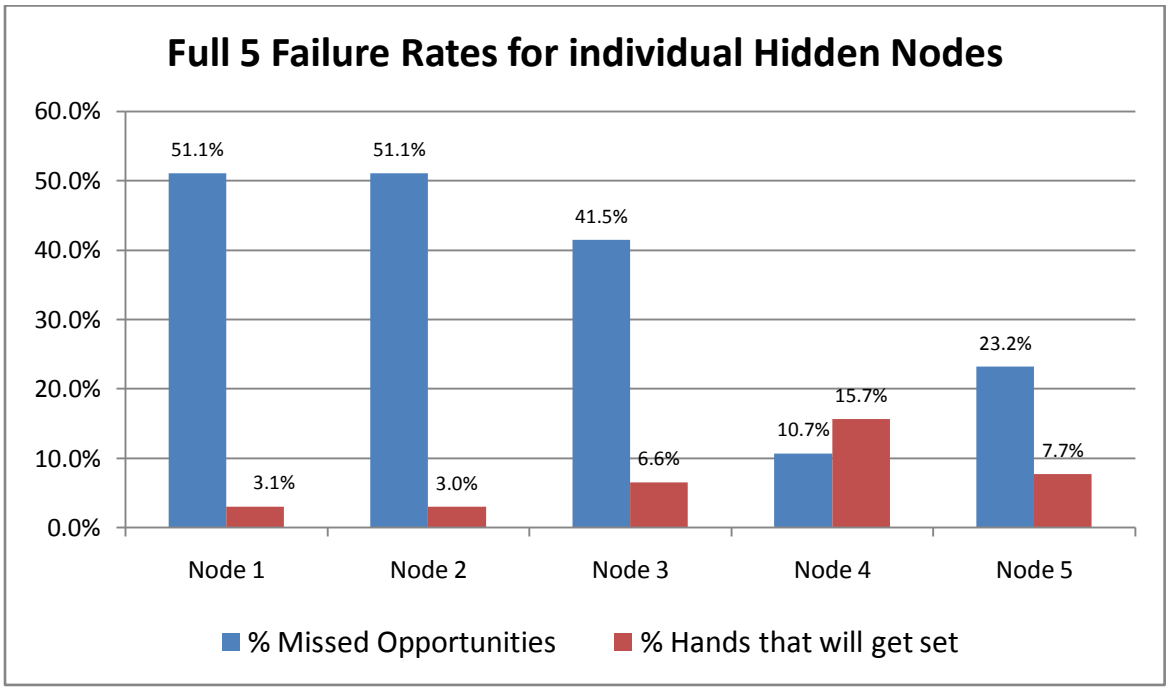

Figure 13

Custom 9 Network

The following graph (Figure 14) shows the network with all weights with rounded to the nearest 5:

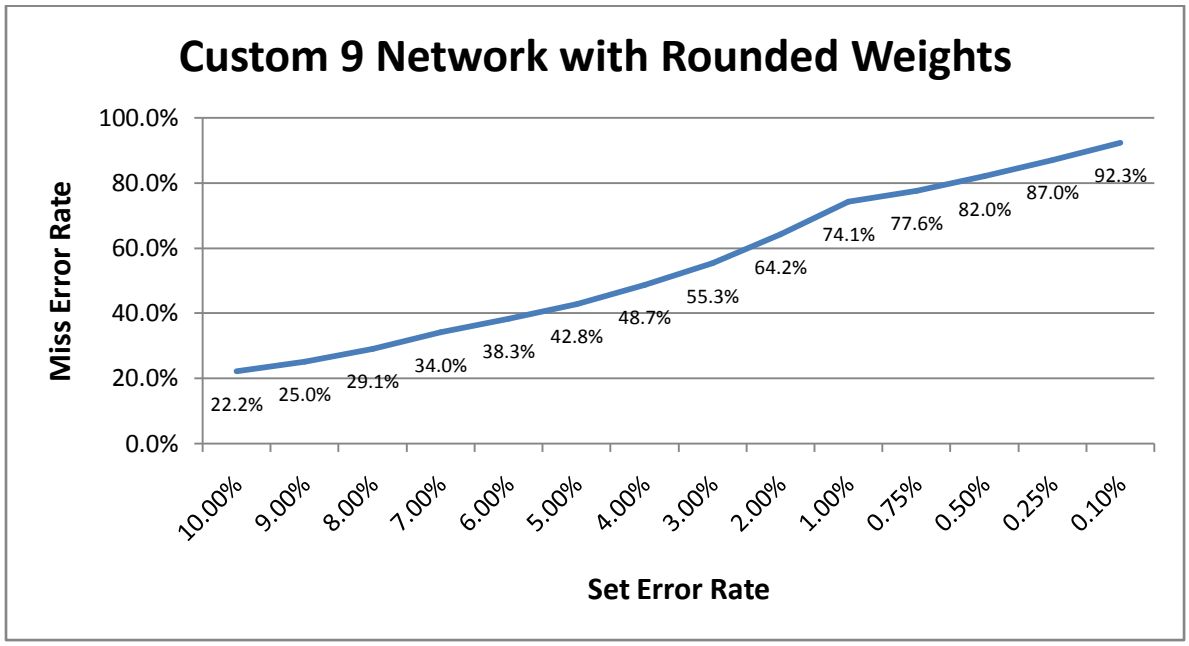

Figure 14 Custom 9 Network with all weights rounded to $\{-40, -35, -30, -25, -20, -15, -10, -5, 0, 5, 10, 15\}$

This graph (Figure 15) shows the individual hidden nodes and their error rates:

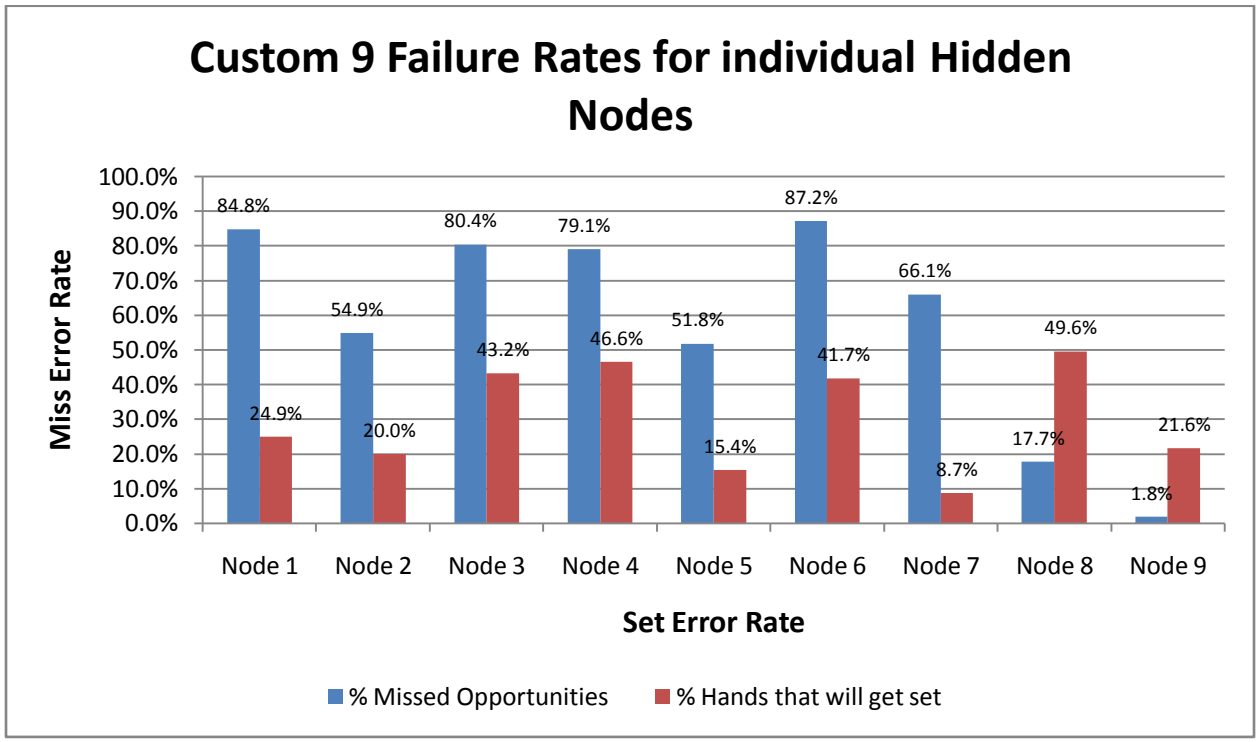

Figure 15

Conclusions

First, I will examine the HCP graphs (Figure 1, Figure 2, Figure 3). It is interesting to note that at the standard HCP values (26 HCP and at least 8 trump), there is only a 0.2% chance of getting set, but 77.1% of hands that can make the contract will never get bid. These values are far from equal; clearly, Bridge players prefer a system that keeps the set error rate very low and are less concerned with the miss error rate. Because the output node of a neural network is a real value between zero and one, it is a simple matter to adjust the cutoff of what is a success or failure to produce a given set error rate. Thus, the graphs of network performance are all shown as one error rate versus the other.

Performance

The High Network (Figure 6), the Distribution Network (Figure 9), and the HCP Seeded Network (Figure 10) all have exceptionally high error rates. The common link between these networks is that not all of the original data is used in determining the output. The apparent conclusion is that the missing inputs are vital to determining success; however, HCP is only given the number of trump and the number of jacks, queens, kings, and aces, and it can determine success to a much better accuracy.

Graphs of the remaining networks clearly show that they have been trained to minimize both errors equally rather than weight one as more important than the other, which is a problem because I am looking for a system that will appease Bridge players who want a specifically small set rate error. Joone uses the root mean square error as the error function that it tries to minimize in training, but to get data that has a low miss error rate at the 0.2% set error rate level, it may be necessary to use a function that specifically tries to minimize set errors. It is evident from most of the graphs (especially the Full Networks: Figure 4 and Figure 5) that the networks performs better than standard HCP when the set error rate is allowed to be higher. However, the networks' miss error rates rise drastically as the set error rate is lowered to very small values.

Contrary to expectation, the Full 5 Network (Figure 5) outperforms the Full 50 Network (Figure 4) and the Custom 9 Network (Figure 8) outperforms the Custom 34 Network (Figure 7). The most logical explanation is overtraining. The extra hidden nodes allow the networks to find random relationships among the training inputs that have no causal relation to the output but reduce the root mean square error anyway. When tested on new data, the over trained networks perform worse than those that did not have enough nodes to find these meaningless relationships.

The only network that actually predicts success with better accuracy than the HCP model is the Full 5 Network. However, because the Custom 9 Network performs reasonably well and is another type of network, I use them both in further analysis.

Rule Extraction

As can be seen in the graph of performance with zeroed weights (Figure 11), zeroing has large negative effects on performance. At the 0.2% set error rate level, the miss error rate climbs from 76.4% to 99.1%. No usable rules can be extracted from this data.

Rounded weights prove more useful. Performance after weight rounding is shown for the Full 5 Network and the Custom 9 Network in Figure 12 and Figure 14 respectively. The rounding parameters were chosen to maximize the simplification while minimizing the damage to the network's performance. For example, rounding to the nearest 7 for the Custom 9 Network increases the miss error rate by over 10% (at set error rate = 0.2%) compared to the nearest 5, but rounding to the nearest 3 only decreases it by less than 1%. Although the performance has gotten worse (the HCP model is now slightly better), the weights are greatly simplified, and rule extraction seems possible. However, I am unable to deduce any meaning from the tables of weights (see Table 1 and Table 2 at the end of this document).

Examining individual hidden nodes reveals little useful rule-making information. The weights of the different nodes are fairly similar, and no one node accurately predicts the final output (the weights can be seen at the bottoms of Table 1 and Table 2).

Discussion and Further Research

I did not determine a better system than the HCP system to use for determining accurate contracts, but that does not mean that another does not exist. In fact, there are already known modifications to HCP that increase its accuracy (Rubens, 1969). It is plausible that a system that looks at relationships between hands can predict success at even greater accuracy than currently used methods.

The next step in this research is to build a new neural network engine that does not solely try to reduce the root mean square error, but rather tries to reduce a custom error that can be adjusted to assure a very small set failure error. From there, new network structures as well as ones like the ones I tested here should be trained and tested. The results from this experiment can act as a guide to determining appropriate test network structures. Future neural networks should have fewer than thirty-four hidden nodes (and possibly fewer than that) as too many causes overtraining. Also, the neural networks should have access to the entire input of the two hands; some input values can be set to zero so long as those values are not zero across all hidden nodes.

Another path to pursue would be to use more advanced rule extraction techniques as mine failed to adequately extract a small set of rules.

If a set of information is found that seems suitable, it must be tested and verified. First, it should be tested using a typical decision tree that branches on whether the conditions of the set of information are met or not. Next, it should be tested using a standard Bridge simulation

(single dummy as opposed to Double Dummy). Because Bridge is not a perfect information game, each player's knowledge must be declared; for this simulation, each player would know his hand, the dummy's hand, and the set of information that is being tested. In a real game, that set of information would be made known through the bidding, so in the simulation, the players should know about it. If the experimental information set performs well on these tests (if it continues to accurately place teams in appropriate contracts, then all that is left to be done is the challenging task of creating a new bidding system that can communicate the information effectively enough to get a team to their desired bid.

Appendix of Bridge Terminology

A Round of Bridge

Each player is dealt thirteen cards randomly from a standard deck of 52; this is the player's hand. The round is then split into two phase: bidding and gameplay. During bidding, the players bid, auction style, for the contract of the round. Each subsequent bid raises the contract, and if there are three passes in a row, the last player to bid wins the contract.

Once the bidding is over, the gameplay begins. One of the players who won the contract reveals his hand to the table (he is the dummy); his hand is controlled by his partner (who is known as the declarer). Thirteen tricks are played, and the score is calculated based on the contract decided during the bidding and how many tricks each team won: the contract winning team gains points if they took enough tricks to satisfy their contract, and the opposing team gains points if they did not.

Double Dummy Bridge

Double Dummy (DD) Bridge is a perfect information variant of Bridge useful for simulations. As stated above, in a round of Bridge, one player is declared the dummy and lays his hand on the table for all the other players to see. In DD Bridge, one player from each team is a dummy. Each remaining player knows his own hand and two other hands and can then perfectly predict his opponent's hand. Thus, it is a game of perfect information and can be solved to find the perfect plays using a minimax algorithm.

In the world of Bridge research, there is serious discussion about the accuracy of the Double Dummy result as compared to the actual Bridge result. The reasoning behind it has to do with special scenarios that may arise in Bridge gameplay where, because not all of the information is known, a player may inadvertently make a sub-optimal play that may change the number of tricks a team takes. However, Double Dummy results are often used in studies such as this one as they tend to be a good estimate of how a real hand of Bridge will go (Ginsberg, 2001).

Contract

The highest bid made during the bidding phase becomes the contract for the round. The team that made that bid is under contract to take a certain number of tricks during the gameplay. If your team has the contract, and you take that number of tricks, then you gain points. However, if you do not take enough tricks to satisfy your contract, your opponents gain many points. The penalties for missing the contract (known as "getting set") are severe.

Game Bid

A Game Bid is a contract that one team makes that is so valuable that, if it is successful, that team wins the game. Note that Bridge is played in matches, so although winning a game is important, it does not mean that the players stop playing. One strives to always bid at least as high as one of the possible game bids (there are five) as they provide significantly more points than the bids just below them. This paper looks at the Major Suit Game Bid, a contract where either Spades or Hearts is trump and the team must take ten of the possible thirteen tricks.

Trick

Each trick consists of one player leading a card and the other three players playing one of their cards on it. Thus, each trick is 4 cards, and there are 13 tricks total. The rules of trick taking are not relevant to this paper.

Trump

One suit may be determined as trump (this would happen in the bidding phase). If this is the case, than a player can use a card in that suit to take a trick that a higher card in a different suit would have taken. The specific rules are not necessary for this paper, but the fact that having more trump than your opponents greatly increases your chance of success is.

Rounded Node Weights from the Trivial Network that has 5 Hidden Nodes

Card	Node 1		Node 2		Node 3		Node 4		Node 5	
Contribution from Player:	1	2	1	2	1	2	1	2	1	2
trump	0	0	10	0	0	0	-10	-10	-10	-10
trump	0	0	10	10	-10	0	-10	-10	-10	-10
trump	0	10	0	0	0	10	-10	-10	-10	-10
trump	0	0	10	10	0	0	-10	-10	-10	-10
trump	10	0	10	0	0	0	-10	-10	-10	-10
trump	0	10	10	0	0	0	-10	-10	-10	-10
trump	10	0	0	0	10	10	-10	-10	0	-10
trump	0	10	10	0	10	0	-10	-10	0	-10
10 trump	10	10	0	0	10	0	-20	-10	-10	-20
J trump	10	10	10	10	10	10	-20	-30	-10	-10
Q trump	20	10	20	10	10	10	-30	-30	-20	-10
K trump	20	20	20	20	20	20	-30	-40	-30	-30
A trump	30	30	30	30	30	30	-40	-40	-40	-40
2	-10	-10	-10	-10	0	-10	10	10	10	10
3	-10	-10	-20	-10	0	-10	10	10	10	10
4	-10	0	-10	-10	0	-10	10	10	10	10
5	-10	-10	-10	-10	-10	-10	10	10	0	10
6	-10	-10	-10	-10	-10	-10	0	10	10	10
7	-10	-20	-10	-10	0	0	10	10	10	10
8	-10	-10	-10	-10	-10	0	10	10	10	10
9	-10	-10	-10	-10	0	0	10	10	10	10
10	0	-10	-10	0	0	0	10	10	10	10
J	0	-10	-10	-10	-10	0	10	10	10	10
Q	0	0	-10	-10	10	10	10	0	0	0
K	0	10	10	0	10	10	0	0	-10	0
A	10	20	20	10	20	30	-10	-10	-10	-10
2	0	-10	-10	0	-20	-10	10	10	10	10
3	-10	-10	-10	0	0	-10	10	10	10	10
4	0	-10	-10	0	-10	-10	10	10	10	10
5	-10	-10	-10	0	-10	-20	10	10	10	10
6	-10	-10	0	0	-10	-20	10	10	10	10
7	0	-10	-10	0	-10	-20	10	10	10	10
8	0	-10	-10	0	-10	-10	10	10	10	10
9	0	-10	-10	0	-10	-10	0	10	10	10
10	0	0	-10	-10	-10	-10	10	10	0	10
J	0	-10	0	0	-10	-10	0	10	10	0
Q	10	0	0	0	-10	-10	0	0	0	0
K	20	10	10	10	0	-10	0	0	0	-10
A	20	20	20	20	20	10	-10	-10	-20	-20
2	-10	-10	-10	-10	-10	-10	10	10	0	10
3	-10	-10	-10	-10	-10	-10	10	10	10	10
4	-20	-10	-10	-10	0	0	10	10	10	10
5	-10	0	0	-10	-10	-10	10	10	10	10
6	-10	-20	-10	-10	0	0	10	10	10	10
7	-10	-10	-10	-20	-10	0	0	10	10	10
8	-10	-10	0	-10	0	0	10	10	10	10
9	-10	-10	-10	-10	0	-10	10	10	10	0
10	0	-10	-10	-10	-10	0	10	10	0	0
J	-10	0	-10	-10	-10	0	10	10	0	10
Q	0	-10	0	-10	0	10	0	0	0	0
K	0	0	10	0	10	10	0	0	0	0
A	10	10	20	10	30	30	-10	-10	-10	-10
Biases	-4.94444		-5.51157		-4.65767		1.881697		3.257047	
Weight of Result toward Output	1.893407		1.658424		2.468913		-2.92167		-2.11999	

Table 1

Rounded Node Weights from the Custom Network that has fewer (9) Hidden Nodes

Card	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9	
Contribution from Player:	1	2	1	2	1	1	2	2	1	2
trump	5	0	0	0	5	-5	0	0	-10	-10
trump	0	0	0	0	5	-5	0	-5	-10	-10
trump	0	-5	0	0	5	0	0	0	-10	-10
trump	0	0	0	0	5	-5	0	-5	-10	-5
trump	0	-5	0	0	0	-5	0	5	-10	-10
trump	0	-5	0	0	0	-5	0	0	-10	-10
trump	0	0	0	0	5	-5	0	-5	-10	-10
trump	0	0	0	0	5	-5	0	-5	-10	-10
10 trump	0	-5	0	0	5	-5	0	-5	-10	-15
J trump	0	-5	0	0	5	-5	5	-5	-15	-20
Q trump	0	-5	0	0	10	-10	0	-10	-25	-20
K trump	-5	-5	0	-5	10	-10	10	-5	-35	-30
A trump	-10	-5	-5	-5	15	-10	15	-15	-40	-40
2	0	0	0	0	0	0	-5	5	15	10
3	0	0	0	0	-5	5	-5	5	10	15
4	0	0	0	0	-5	0	-5	5	10	10
5	0	0	0	0	-5	5	-5	5	10	10
6	0	0	0	0	-5	10	-5	5	10	10
7	0	0	0	0	-5	5	-5	5	10	10
8	0	0	0	0	-5	0	-5	5	10	10
9	0	0	0	0	-5	0	0	0	10	10
10	0	0	0	0	0	0	-5	5	10	5
J	0	0	0	0	-5	5	-5	-5	5	10
Q	0	0	0	0	-5	5	0	0	0	5
K	0	0	0	-5	0	0	5	0	-5	-5
A	0	0	-5	-5	5	-10	10	-15	-15	-15
2	0	0	0	0	-10	5	-5	5	10	10
3	0	0	0	0	-5	5	-5	0	10	10
4	0	0	0	0	-5	5	-5	0	10	10
5	0	0	0	0	-5	5	-5	5	10	10
6	0	0	0	0	0	5	-5	0	10	10
7	0	0	0	0	-5	5	-5	5	10	10
8	0	0	0	0	-5	5	-5	0	10	10
9	0	0	0	0	-5	5	-5	5	10	10
10	0	0	0	0	-5	0	-5	0	10	10
J	0	0	0	0	-5	5	0	5	5	5
Q	0	0	0	0	0	0	0	5	5	0
K	0	0	0	0	5	-5	5	-10	-5	-5
A	0	0	-5	-5	5	-5	10	-10	-15	-15
2	0	0	0	0	-5	0	-5	0	10	10
3	0	0	0	0	-5	5	-5	5	10	10
4	0	0	0	0	0	5	-5	5	10	10
5	0	0	0	0	-5	5	-5	5	10	10
6	0	0	0	0	-5	0	-5	0	10	10
7	0	0	0	0	-5	5	-5	5	10	10
8	0	0	0	0	-5	5	-5	5	10	10
9	0	0	0	0	-5	0	-5	5	10	10
10	0	0	0	0	-5	5	-5	5	5	10
J	0	0	0	0	-5	5	0	0	5	10
Q	0	0	0	0	-5	5	0	0	0	0
K	0	0	-5	-5	5	0	5	-5	-5	-5
A	0	0	-5	-5	5	-10	10	-10	-15	-15
Biases									-4.94444	
Weight of Result toward Output									1.893407	

Table 2

Works Cited

Butler, B. (2008, January 23). *Bridge Probabilities*. Retrieved April 2008, from http://www.durangobill.com/Bridge.html

Ginsberg, M. L. (2001). GIB: Imperfect Information in a Computationally Challenging Game. *Journal of Artificial Intelligence Research 14* , 303-358.

JAMA: Java Matrix Package. (2005, July 13). Retrieved April 2008, from http://math.nist.gov/javanumerics/jama/

Marrone, P. (2008, January 21). *Joone - Java Object Oriented Neural Engine*. Retrieved April 2008, from http://www.jooneworld.com

Rubens, J. (1969). *The Secrets of Winning Bridge.* New York: Grosset and Dunlap.