Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games (CIG 2007)

Bridge Bidding with Imperfect Information

Lori L. DeLooze United States Naval Academy
572M Holloway Rd
Annapolis, MD 21402
delooze@usna.edu

James Downey
University of Central Arkansas
201 Donaghey Avenue
Conway, AR 72035
downey@uca.edu

***Abstract*—Multiplayer games with imperfect information, such as Bridge, are especially challenging for game theory researchers. Although several algorithmic techniques have been successfully applied to the card play phase of the game, bidding requires a much different approach. We have shown that a special form of a neural network, called a self-organizing map (SOM), can be used to effectively bid no trump hands. The characteristic boundary that forms between resulting neighboring nodes in a SOM is an ideal mechanism for modeling the imprecise and ambiguous nature of the game.**

Keywords : Bridge, Self-Organizing Map, Kohonen, Bidding

I. Introduction

Game theory is a particularly rich area for study. Many researchers deal with two-player games, such as chess or checkers, where each opponent is presented with full information. A greater challenge, however, is multiplayer games with both incomplete information and an element of chance, such as Poker or Blackjack. The game of Bridge falls in between these two extremes. It is a multiplayer game, with opposing teams and incomplete information, but the only element of chance involved is the initial randomness in the deal.

Several algorithmic approaches have proven somewhat successful with multiplayer imperfect information games. [1] Given the limited information provided, the missing information is inferred. Generally, a Monte-Carlo sampling technique generates a set of representative hands the opponents may have. A standard minimax algorithm selects the most likely holding and makes a corresponding move. The model is update as additional information becomes available. Eventually converging on an acceptable solution. In Bridge, however, we are looking for the ideal solution rather than just an acceptable solution.

The dealer distributes 13 cards from a standard 52-card deck to each of four players who have been named according the compass directions (North-South against East-West). The game consists of two activities, the bid and the play of the cards. Commercial products such as Bridge Baron, GIB[2] and Jack, the World Computer Bridge champion, have proven to be especially effective in the play of the cards. Bidding, however, has shown to be a more complex problem.

Bidding is a conversation between two cooperating team members against an opposing partnership. Each partnership uses an established bidding system to exchange information and interpret their partner's bidding sequence. Each player only has knowledge of their own hand and any previous bids. Bidding begins with the dealer and ends with a legitimate bid followed by three sequential passes. The highest bid becomes the contract. A Bridge contract consists of a suit (or no trump) and a level. The level means the number of tricks over a standard "book" of 6 tricks. Teams are awarded bonus points if they bid and make "game" (3NT, 4♥, 4♠, 5♣, or 5♦) or "slam" (12-13 total tricks).

Once the final contract is reached, the opposing team lays down an initial card and the play phase of the game begins. Each player must present one card for each of the 13 tricks. Players must all follow suit, if they have that suit, or play an alternate suit if they don't. In a no trump contract, the highest card of the suit led takes the trick. In a trump contract, in contrast, the highest card of the trump suit takes precedence over all others. The contract indicates a guess as to the number of tricks the team can take.

The scoring depends on the number of tricks taken and the final contract. Points are scored for that team if they make or exceed their contract or given to the other team if they fail in their attempt. Additional points are granted based on "vulnerability" of the team who wins the contract. Point values vary depending scoring system used and the number of competing tables. The IMP method awards points based on the arithmetic difference between scores according to a standard conversion table. The MP method, in contrast, gives 2 points for each score worst that the pair's score, 1 point for each equal score, and 0 points for each better score. The winner is determined by the total points at the end of a finite number of rounds.

Because the biggest differentiator between Bridge-playing ability is the quality of the bidding, we will focus on creating an effective method for computer bidding using an artificial neural network. Cognitive studies have shown that human performance in Bridge can be attributed to the acquisition of high-level patterns and chunks of knowledge gained through experience. Frank, Bundy and Basin [3] showed that standard minimax may be applicable to the card play portion of the game, but fails to extend to the bidding phase of the game. Other models and algorithms needed to be developed for this complex problem.

Bridge bidding can be reduced to a multifaceted conversation between partners. As such, it would seem natural reduce it to its semantic and pragmatic elements [4]. Each bid can be classified as one of four common acts:

1. Asserting. "By making this bid, I assert that my hand has these properties"
2. Denying. "By making this bid, I deny that my hand has these properties"
3. Asking and Answering Questions. "If your hand is of type 1, make bid 1; if it is of type 2, make bid 2, etc. (e.g. asking how many aces your partner has)
4. Interrupting. "The primary purpose of this bid is to stop the opponents communicating

Another approach that has shown some promise is a neural network [5]. Neural networks take the raw input data and construct appropriate outputs by successively recalculating the weights on the connections between their nodes. Some of the input nodes in a network for contract bridge would include preprocessed values such as high card points and suit length. Although these other approaches have used neural networks in conjunction with other artificial intelligence techniques, a special form of neural network is showing some promise.

II. SELF-ORGANIZING MAPS

The Self-Organizing Map, also called a Kohonen Map, is one of the most prominent artificial neural network models adhering to the unsupervised learning model [6]. The model consists of many neural processing units. Each of the units is assigned a multi-dimensional weight vector, *mi*. The weight vectors have the same dimensionality as the input patterns. Training self-organizing maps involves input pattern presentation and weight vector adaptation. Each training iteration starts with the random selection of one input pattern. The self-organizing map examines this pattern and decides each unit's activation.

Usually, the Euclidean distance between weight vector and input pattern is used to calculate a unit's activation. The unit with the lowest activation becomes the winner of the training iteration. Finally, the weight vectors of the winner as well as the weight vectors of selected units around the winner are adapted. This adaptation results in a gradual reduction of the component-wise difference between input pattern and weight vector. The model generally consists of a two-dimensional neuron arrangement (map), as shown in Figure 1, though topologies of higher dimensions are also conceivable.

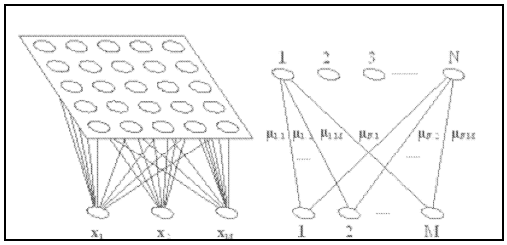

Figure 1: Self-Organizing Map

Each neuron has a representative set of M features, called a vector. During the training process, the feature weights are modified according to the input signal and the neurons proximity to the winning neuron. Each weight is increased or decreased to more closely resemble the matching the input vector, with neurons closer to the winning neuron making greater changes in the weights than those further away. Because of the algorithm, an organized network develops where similar input patterns are arranged with a degree of proximity between the locations of excited neurons. The neurons are arranged by the input patterns by neighborhoods. That is, the neurons not are adapted individually, but with neighboring neurons.

Unlike many other types of networks, a Self-Organizing Map does not need a target output to be specified. Instead, the area of the lattice where the node weights match the input vector are selectively optimized to more closely resemble the data for the class of the input vector. From an initial distribution of random weights, and over many iterations, the SOM eventually settles into a map of stable zones. Each zone is effectively a feature classifier, so you can think of the graphical output as a feature map of the input space. Any new, previously unseen input vectors presented to the network will stimulate nodes in the zone with similar weight vectors.

Several variations of the Kohonen algorithm exist [7]. The algorithm used for the SOMs in this research is as follows:

1. Initialize each node's weights.
2. Choose a vector at random from the set of training data and present it to the lattice.
3. Examine every node and determine which one's weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).
4. Calculate the radius of the neighborhood of the BMU. This is a value that starts large, typically set to the 'radius' of the lattice, but diminishes each time-step. Any nodes found within this radius are deemed to be inside the BMU's neighborhood.
5. Adjust the weights of each neighboring node to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered.
6. Repeat step 2 for N iterations.

One method to determine the best matching unit is to iterate through all the nodes and calculate the Euclidean distance between each node's weight vector, W_i, and the current input vector, V_i. The node with a weight vector closest to the input vector is tagged as the BMU. The Euclidean distance is given as:

$$dist = \sqrt{\sum_{i=0}^{i=n} (V_i - W_i)^2}$$

A unique feature of the Kohonen learning algorithm is the area of the neighborhood shrinks over time. This is accomplished by making the radius of the neighborhood shrink over time. To do this, we use the exponential decay function:

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\lambda}\right) \quad t = 1,2,3 \ldots$$

where σ_0, stands for the width of the lattice at time t_0, λ denotes a time constant, and t is the current time-step (iteration of the loop). Over time the neighborhood will shrink to the size of just one node – the BMU. After fixing the radius, we iterate through all the nodes in the lattice to decide if they lie within the radius and adjust the weights accordingly. Every node within the BMU's neighborhood (including the BMU) has its weight vector adjusted according to the following equation:

$$W(t+1) = W(t) + L(t)(V(t) - W(t))$$

where t represents the time-step and L is a small variable called the *learning rate,* which *decreases with time*. The decay of the learning rate is calculated each iteration using the following equation:

$$L(t) = L_0 \exp\left(-\frac{t}{\lambda}\right) \quad t = 1,2,3 \ldots$$

The learning rate at the start of training is set to some constant and then gradually decays over time so during the last few iterations it is close to zero. The effect of the learning should decrease proportionally according to the distance of the node from the BMU. In fact, the edges of the BMU's neighborhood should have barely any effect at all. Ideally, the learning should fade over distance according to the Gaussian decay shown in Figure 2.

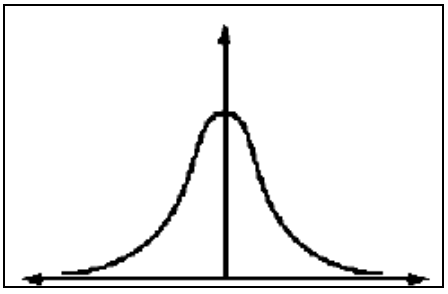

Figure 2: Gaussian Decay Around BMU

To achieve this, all it takes is a slight adjustment to the equation above.

$$W(t+1) = W(t) + \Theta(t)L(t)(V(t) - W(t))$$

where θ, represents the influence a node's distance from the BMU has on its learning. $\theta(t)$ is given by

$$\Theta(t) = \exp\left(-\frac{dist^2}{2\sigma^2(t)}\right) \quad t = 1,2,3 \ldots$$

where *dist* is the distance a node is from the BMU and σ, is the width of the neighborhood function. Note that θ also decays over time. Geometrically speaking, the weight vectors of the adapted units are moved a bit towards the input pattern. The amount of weight vector movement is guided by a learning rate decreasing in time. The number of units that are affected by adaptation is determined by a so-called neighborhood function. This number of units also decreases in time. This movement has as a consequence. The Euclidean distance between those vectors decreases and thus, the weight vectors become more similar to the input pattern.

The respective unit is more likely to win at future presentations of this input pattern. The consequence of adapting not only the *winner* alone but also a number of units in the neighborhood of the *winner* leads to a spatial clustering of similar input patterns in neighboring parts of the self-organizing map. Thus, similarities between input patterns that are present in the multi-dimensional input space are mirrored within the two- dimensional output space of the self-organizing map.

The training process of the self-organizing map describes a topology-preserving mapping from a high-dimensional input space onto a two-dimensional output space where patterns that are similar in terms of the input space are mapped to geographically close locations in the output space.

III. EXERIMENTAL SETUP

When setting up the initial training vector of the self-organizing map, we decided that the only information available to the bidder is the layout of his hand and the current bidding history. Therefore, the input vector is a series of discrete values to show the distribution of cards in each suit and quality of the cards. Generally, Bridge players value cards according to rank with ace = 4 points, king = 3 points, queen = 2 points and jack = 1 point. Although some players also add or subtract points based on length or shortness in a particular suit, we will ignore that factor because it will be captured in the card distribution.

As with other attempts at using machine learning for Bridge bidding [8], the first step is to produce training examples. We generate a set of training instances that represent card distribution by suit and the total number of high card points (HCP) as described above. Each distribution is then mapped to an appropriate bid according to the guidelines published by the American Contract Bridge League [9]. Generally a Bridge contract can be determined in four phases: the opening bid, the responder's response to the opening bid, the opener's response and, finally, the responder's final placement of the contract. Although some card distributions benefit from additional communication, we will limit the bidding history to these four phases.

The bidding history is necessary to interpret previous bids. We have determined only 6 SOMs are required for an accurate bidding scheme: opening bids, no trump responses, primary responses, overcalls, re-bids by opener, and re-bids by responder. The first SOM, the opening bid SOM (Figure 3), does not require any information on any previous bidding history because these are based solely on the construction of the hand under consideration. Opening bids fall into three general categories: no-trump, major suit (hearts and spades) and minor suits (clubs and diamonds). The remaining SOMs are constructed to respond successively to the opening bid.

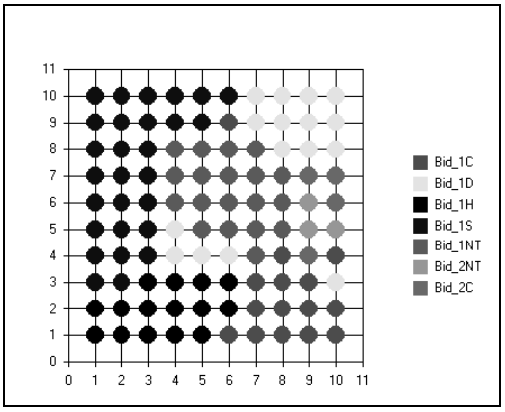

Figure 3: Opening Bids

To test the effectiveness of this bidding mechanism, we will separate the auction from the play and stage a small tournament between four teams at two tables. One table will have two players sitting East-West bidding with our system, designated as BridgeSOM, and North-South bidding with Jack. The other table will have North-South bidding with BridgeSOM and East-West bidding with Jack. The two tables will play the same set of 24 boards. Once the bidding is complete, Jack will play the cards for all four players.

A. No Trump Hands

Although BridgeSOM is able to bid on all randomly distributed Bridge hands, we will simplify this initial evaluation by using a special type of bridge hand that can be played using very well defined rules. A No-trump hand has 15-17 High Card Points and is balanced. High Card Points are simply the sum of the accumulated card ranks. Consider the following hand:

♠KT ♥AKQ3 ♦J63 ♣K864

It has a total of 16 HCP spread among the four suits (3 in spades + 9 in hearts + 1 in diamonds + 3 in clubs). In addition, this hand is balanced which means that it has all four suits distributed in a 3-3-3-4 configuration or with only one card changed from that (ie, 2-3-3-5 or 4-2-3-4).

A computational scheme such as a self-organizing map is ideal for Bridge because of the fuzzy boundaries between sets. Note the Gaussian decay around each BMU in the SOM, Figure 4. This means that adjacent nodes will match both patterns to some degree, but the node will be labeled with the pattern that matches best. If you ask any Bridge player about a particular hand, they may be able to describe two or more possible responses. These are the hands that fall on these boundaries.

Mathematicians love to play Bridge because there are some very logical guidelines for bidding and play. For example, guidance suggests that a partnership needs a total of 25 points for a game in no trump (3NT), 26 points for a game in either hearts or spades (4♥or 4♠) and 29 points for a game in diamonds and clubs (5♦ and 5♣).

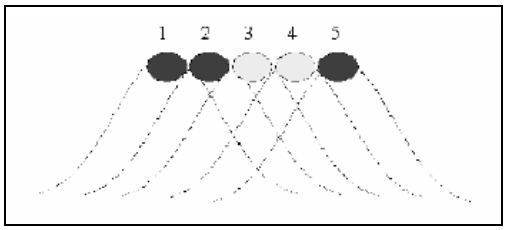

Figure 4: Neighboring Neurons

If the first bidder sees a balanced hand with 15-17 points, they should open the auction with 1 NT. Recall that a total of 25 points is needed for a game in no trump. Therefore, the responder (opener's partner) should respond according to the following rules:

1. With 0-7 points: Pass

2. With 8-9 points: bid 2NT (opener will go to game, 3NT, with 17 points for a total of 25 points or pass and stay at 2NT for the contract.

3. With 10-15 points: bid 4NT, which asks the partner to pass with 15 points, bid 5NT with 16, bid 6NT with 17.

These rules are not programmed as a set of conditionals. Instead, we create a set of training vectors that reflect these possible configurations and label them with the appropriate responses. The input vector consists of 5 values: the number of cards in each suit and the total number of high card points.

For this evaluation, we will use the initial opening bid SOM and the no trump response SOM, Figure 5. If there are any opening bids other than the expected 1 NT bid, they will be handled with the appropriate SOMs until a contract is reached. The SOMs are created using representative training sets well before the tournament. Getting a bid is almost instantaneous, as we are simply finding the node that is the best match to the current player's card distribution, HCPs and bidding history.

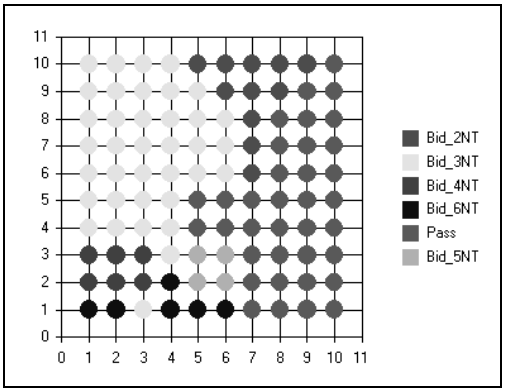

Figure 5: No Trump Responses

B. No Trump Results

We tested the performance of BridgeSOM against Jack in a match of 24 boards. Although the match was performed and scored the same way as an official bridge match, all the boards were configured for an initial 1NT bid, with the same dealer each hand and no vulnerability designated for either side. We turned off all special bidding features for Jack. Neither Jack nor BridgeSOM made all bids as expected.

Table 1 indicates the initial bid, the resulting final contract, and final result for both Jack and BridgeSOM when opening the bidding with identical hands.

Table 1
Jack vs BridgeSOM

Board	Jack	BridgeSOM
1	1NT/2NT/2NT	1NT/2NT/2NT
2	1NT/4NT/4NT	1NT/3NT/4NT
3	1NT/2NT/2NT	1C/2NT/2NT
4	1NT/3NT/3NT	1NT/4NT/3NT
5	1NT/3NT/3NT	1NT/3NT/3NT
6	1C/2H/2H	1NT/2NT/2NT
7	1NT/3NT/3NT	1NT/3NT/3NT
8	1NT/2NT/3NT	1NT/2NT/3NT
9	1NT/2NT/2NT	1H/2H/2NT
10	1NT/3NT/3NT	1NT/4NT/3NT
11	1NT/2NT/3NT	1NT/3NT/3NT
12	1NT/3NT/3NT	1NT/3NT/3NT
13	1H/3H/3H	1NT/3NT/2NT
14	1NT/3NT/2NT	1NT/3NT/2NT
15	1NT/2NT/3NT	1NT/3NT/3NT
16	1NT/3NT/2NT	1NT/2NT/2NT
17	1NT/3NT/4NT	1NT/3NT/4NT
18	1NT/3NT/3NT	1S/3S/3S
19	1NT/3NT/3NT	1NT/3NT/3NT
20	1NT/4NT/4NT	1NT/4NT/4NT
21	1NT/3NT/3NT	1NT/3NT/3NT
22	1NT/2NT/2NT	1NT/1NT/2NT
23	1NT/3NT/5NT	1NT/3NT/5NT
24	1NT/2NT/2NT	1NT/2NT/2NT

Points were awarded for the differences between final board scores according to table 2.

Table 2
IMP Table

Point difference		IMPs	Point difference		IMPs	Point difference		IMPs
from	to		from	to		from	to	
0	10	0	370	420	9	1750	1990	18
20	40	1	430	490	10	2000	2240	19
50	80	2	500	590	11	2250	2490	20
90	120	3	600	740	12	2500	2990	21
130	160	4	750	890	13	3000	3490	22
170	210	5	900	1090	14	3500	3990	23
220	260	6	1100	1290	15	4000 or more		24
270	310	7	1300	1490	16			
320	360	8	1500	1740	17			

Source: Duplicate Bridge, wikipedia.com

Jack won the tournament with 22 IMPs to BridgeSOM's 17 IMPs. There were only 6 boards that presented significant differences between the two systems: Boards 10, 11, 13, 15, 16, and 18. Of these, board 10 was the most important. If the bidding had been different on this one board, BridgeSOM could have won the tournament. The bidding began the same with both systems. Jack however, ended up with a 3NT game contract while BridgeSOM ended up a level higher at 4NT. Because of the unlucky distribution of the cards, the results were, in fact, just 3NT.

When we later examined the actual node that caused the 4NT bid rather than the alternative 3NT bid, it was on the boundary between the two. The distribution and card values for this hand, therefore, could have supported either a 3NT or 4NT bid. If the cards had been distributed differently in the opponents' hands, we could both have both made 4NT. This would have ended the tournament at 11 to 17, in our favor.

IV. CONCLUSION

Self-organizing maps and other computational intelligence methods are ideal for games with incomplete information. They are tolerant of imprecision, uncertainty and partial information. Neural networks allow a degree of imprecision in the data used to train the nets without a great impact on the learning. Our Self-Organizing Map was trained with a minimal subset of the training data, yet is quite capable of operating with immense data variability because perfect discrimination between bidding options is not required.

We have shown that a combination of two self-organizing maps can be used to find an optimal strategy for no trump Bridge hands. Although this uses only assertion, one of the four semantic and pragmatic elements of a bidding conversation, we can use similar techniques to model denial, asking questions, and interrupting.

REFERENCES

[1] N. Sturtevant, "A.Comparison of Algorithms for Multi-Player Games," Lecture Notes in Computer Science, Springer, Berlin, 2003.

[2] M. Ginsburg, "GIB: Imperfect Information in a Computationally Challenging Game," *Journal of Artificial Intelligence Research* 14, (303-58), 2001.

[3] I. Frank, D. Basin and A. Bundy, "Combining knowledge and search to solve Single-suit Bridge," *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (195-200), 2000.

[4] B. Gamback, M. Rayner and B. Pell, "Pragmatic Reasoning in Bridge", Technical Report No 299, University of Cambridge, April 1993.

[5] B. Gambäck and M. Rayner. "A Micro-world for Reasoning about Communicating Agents,"*Proceedings of the 9th Annual Workshop and Meeting of the Swedish Artificial Intelligence Society*, (35-39) Stockholm, Sweden, April 1990.

[6] M. Dittenbach, A. Rauber and D. Merkl, "Uncovering the Hierarchial Structure in Data Using the Growing Hierarchical Self-Organizing Map," *Neurocomputing*, 58 (4) (199-216) 2002.

[7] T. Kohonen, Self-Organizing Maps and Associative Memory. Berlin: Springer 1995.

[8] A. Amit and S. Markovitch. "Learning to Bid in Bridge," *Machine Learning*, 63 (287-327) 2006.

[9] A. Grant. *Bidding*. American Contract Bridge League, Memphis, TN, 1990.