Carnegie Mellon University
Research Showcase @ CMU

Dietrich College Honors Theses | Dietrich College of Humanities and Social Sciences

4-23-2009

Opponent Modeling in Texas Holdem with Cognitive Constraints

Jeffrey Sander
Carnegie Mellon University, jsander@andrew.cmu.edu

Follow this and additional works at: http://repository.cmu.edu/hsshonors

Recommended Citation
.

This Thesis is brought to you for free and open access by the Dietrich College of Humanities and Social Sciences at Research Showcase @ CMU. It has been accepted for inclusion in Dietrich College Honors Theses by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

Opponent Modeling in Texas Holdem with Cognitive Constraints

Jeffrey Sander, Christian Lebiere
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15289
jsander@andrew.cmu.edu, cl@cmu.edu

Abstract

The goal of our ongoing project is to develop and test various methods of modeling one's opponent in adversarial domains. The texas hold'em domain will be used because it has the key features of imperfect information, stochasticity, and the ability to deceive. These three components allow for a very wide range of human behavior making it difficult to create feasible models using traditional artificial intelligence techniques. It is precisely in these situations where models based on human cognition are most helpful. Their adaptive and flexible methods can overcome the uncertainty and deceptiveness intrinsic to human behavior aiding with naturally difficult domains. In other words, to understand a human it is helpful to think like one.

Prior Work

To manage the complexity of the Texas Hold'em domain it was necessary to begin with a simpler task. The interaction between a pitcher and a batter in baseball was used for this. Here it was the pitcher's job to predict what action the batter would take based on prior behavior. The pitcher would do this by continually reinforcing strategies consistent with the batter's behavior. Overtime this would allow for the strategy most consistent with the batter's behavior to emerge.

The framework for our work in Texas Hold'em is based on the annual AAAI poker competition. Here a set of agents with various design paradigms are submitted to compete in a heads-up round robin tournament. However, to our knowledge none of these agents are based on cognitive principles and generally use fairly intricate algorithms involving statistical analysis and meta-reasoning. While these approaches may be effective they are hardly cognitively plausible and tell us little about the merits, deficiencies, and intricacies of our own behavior. Additionally, it is difficult for these machine learning paradigms to accurately model human behavior. However, the software used for the tournament was both opensource and available for free download, ideal for our purposes, so we have adapted it to suit our needs.

Texas Holdem

Texas Holdem is a variant of poker sharing with it the elements of chance, strategy, and deception. For our purposes a head-to-head version was designed in which two players face each other for a series of hands. Each player is given a bankroll which carries over from one hand to the next so a player's winnings and losses are accumulated over the course of the match. A player is free to use their bankroll, which in essence represents their up-to-date performance, when deciding on betting, but currently none of the agents do. Both players start with 0 chips and have no upper or lower limit on their bankroll and their bankroll does not limit what they can bet. Consequently, a player with a negative number of chips may still bet. This is not that unrealistic, since outside of tournament play

professional players will tend to have standard betting strategies, independent of their bankroll. The game is zero sum, as any chips which one player wins the other loses. At the end of a match, which consists of a set number of hands, the winner is the player with the most chips.

Each hand consists of a sequence of four rounds, each of which begins with additional cards being dealt followed by betting. The four rounds are the preflop, flop, turn, and river. During the preflop both players are dealt two cards face down. These are called hole cards or the player's pocket and are the only cards which a player will receive individually. All other cards are community cards and will be placed face up on the table and used by both players. Betting follows the deal and the player in seat #1 bets first. Each hand the players' seats will alternate so the player who bets first is the one who previously bet second.

The second round is the flop which begins with three (3) cards being placed on the table face up and ends with a round of betting. This is followed by the turn and river which both begin with a new community card being placed on the table and also end with a round of betting. The player who bet second preflop bets first for all subsequent rounds.

At the end of the river is the showdown. Here both players reveal their cards and whomever has the better hand wins the pot. It is also possible that play never reaches the showdown if a player folds. If this is the case neither player will reveal their cards and the player who did not fold will take the pot.

AAAI Competition & Implementation

Perhaps the best example of recent AI work done in Texas Holdem is the AAAI Computer Poker Competition. Every year entries are submitted from groups around the world which compete against one another in a round robin tournament. Each entry is a bot which connects to a server that runs the match. STATUS messages are continually sent from the server to both of the competing bots which communicate the current state of the match. Each STATUS is a string of text of the form:

MATCHSTATE:<seat #>:<hand #>:<betting>:<cards>

The <cards> portion of the STATUS string takes the form of <seat0's pocket | seat1's pocket/flop/turn/river>, where only the portion of the cards which the player can see is shown. For instance the seat0's pocket portion of the <cards> element won't be shown to the player in seat 1 until the showdown.

Below are a few examples and explanations of STATUS messages:

MATCHSTATE:0:5::Kh4d
indicates that the message is for seat0, is at the beginning of the 5th hand of the match (there aren't any bets yet), and that seat0's pocket cards are a king of hearts and 4 of diamonds.

MATCHSTATE:0:35:rc/crc/:3sJc|/QhTcKc/8c
This message is also for seat0 and occurs at the beginning of the turn (3rd round) for the 35th hand. The betting went as follows:
preflop: seat1 raised, seat0 called
flop: seat0 called, seat1 raised, seat0 called
The cards are as follows:
seat0's pocket: 3 of spades, jack of clubs
flop: queen of hearts, ten of clubs, king of clubs
turn: 8 of clubs

MATCHSTATE:0:35:rc/crc/crc/crc:3sJc|Qs9s/QhTcKc/8c/3d
This is the showdown from the same hand. It occurs after all betting is concluded and both players reveal their cards. The notable difference here is that both seat0's and seat1's cards can be seen in the STATUS message. Also note that this would not be the case if the hand ended with one of them folding in which case cards would remain hidden.
seat0's pocket: 3 of spades, jack of clubs
seat1's pocket: queen of spades, 9 of spades

After a bot/player receives a STATUS message from the server it can reply with an action appended to the end of the STATUS message, for instance:

MATCHSTATE:0:35:rc/crc/:3sJc|/QhTcKc/8c:r
indicates that seat0 would like to begin the 3rd round of betting with a raise.

For limit poker the three possible actions are raise (r), call (c), and fold (f).

Our adaptation of the AAAI Competition makes use of their server with a few marginal differences. We have created a series of agents capable of connecting to and communicating with the server in order to face each other in head-to-head competitions.

A java based texas holdem hand evaluator was used (Davidson, Billings, Papp). It was accessed from lisp using allegro's jlinker foreign function interface.

Modeling Techniques

The Opponent Model of poker behavior was developed using ACT-R, a cognitive architecture. This was done to restrict our model with cognitive constraints. These constraints are embedded in the cognitive architecture which is in abstract a set of empirically derived rules and procedures. These theories are then implemented in software as a testbed for building cognitive models. The relationship between a model and its underlying architecture is similar to that between a computer program and the architecture it is build upon. The architecture specifies the basic components of the system such as memory, instruction set, peripheral devices, as well as how these components interact. The same is true for a cognitive architecture.

Below is a diagram traditionally used to illustrate the components of ACT-R and the relationships between them.

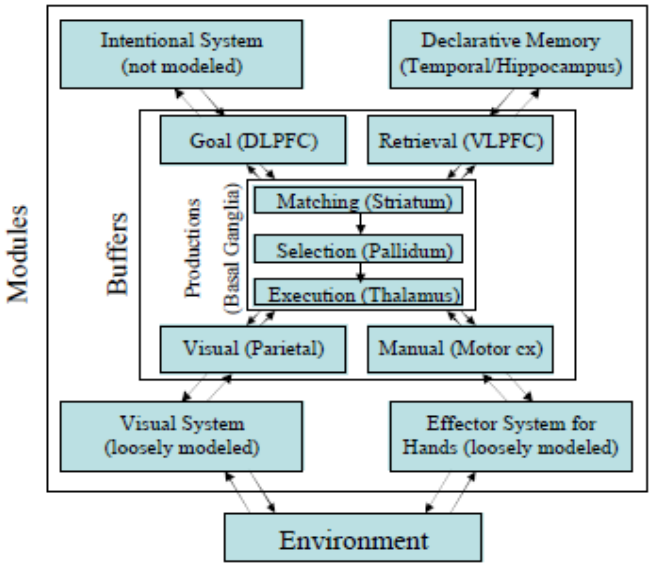

Figure 1: ACT-R 5.0 Architecture

Each component is mapped to a region of the brain based on empircal fMRI data and corresponding data synthetically generated by ACT-R. A primary purpose of this mapping is to validate one of the architecture's core assumptions of approximating true cognitive processes.

Our models primarily make use of the retrieval and declarative memory modules in ACT-R. The declarative memory module simulates long term memory and the retrieval module our ability to retrieve information from it.

ACT-R stores information in chunks, each of which represents a concept or grouping of associated information. Each piece of information stored in a chunk is placed in a slot. Each chunk is also given a type to specify what kind of information it contains. Below is an example of a 'grandmother chunk':

```
grandmother
    isa person
    name 'Grammy'
    age 'undisclosed'
    location 'florida'
    husband 'Gramps'
```

The name of the chunk is grandmother, the type is person, the slots are name, age, location, husband containing 'Grammy', 'undisclosed', 'florida', and 'Gramps' respectively. A model's chunks are kept in declarative memory and may be retrieved by the retrieval module. Retrieval's are done to access a piece of information stored in a chunk's slot. Which chunk is retrieved is specified by the values of the other slots in the chunk. Below is the form a retrieval request normally takes:

```
+retrieval>
    isa person
    name 'Grammy'
    age 'undisclosed'
    husband 'Gramps'
```

Assuming that the grandmother chunk is the only chunk which matches this retrieval request it will be retrieved. Once a chunk is retrieved, the values of slots which weren't specified in the retrieval request can be accessed as shown below:

```
=retrieval>
    location =location
```

If more than one chunk in declarative memory matches a retrieval request than the chunk with the

greater activation will be retrieved. Each chunk is given an activation which represents its strength in long term memory. The activation for a particular chunk is computed given how many times it has been referenced and how long ago these references occurred. Here a reference is either a retrieval of the chunk or the creation of a new chunk with identical slot values. Activation decays over time so the longer ago a reference occurred, the less it will contribute to the chunks total activation. Below is the equation for computing the activation of a chunk:

$$\ln(\sum_{j=1}^{n} t_j^{-d})$$

Here n is the number of times the chunk has been retrieved, t_j the time since the jth retrieval, and d the constant decay parameter, which defaults to 0.5 in ACT-R. As can be seen the activation of a chunk is then the logarithm of the summation of the contributions of each retrieval, where each retrieval contributes activation according to one over the square root of the time since it occurred. It should be noted that the decay parameter does not constitute an additional degree of freedom since it is traditionally set to be 0.5 and for our uses has been left at this default value.

The chunks in the ACT-R opponent model for poker take the form of:

```
s1hand17
        isa s1hand
        preflop-bets  "cc"
        postflop-bets  ""
        turn-bets  ""
        river-bets  ""
        opponentodds  ½
```

There are two types of hands, s1hands and s0hands. These correspond to the two different seats in a heads-up game, but for practical purposes just specifies which player the chunk's opponentodds slot refers to.

The next four slots in the chunk refer to the actions taken during the four rounds of betting. The opponentodds slot reflects the model's estimate of the opponent's odds of winning given the betting that has occurred so far. Because this need not be at the end of a hand, but at the beginning of any of the four rounds, some of the betting slots may be blank.

As an example of this, the above chunk predicts that after the first round of betting the player in seat 0, seat 1's opponent, will have a hand which will win ½ of the time. This is inline with intuition which suggests the same thing given that both players have called.

A model functions by storing four chunks in declarative memory at the end of each hand. There is one chunk for each round of betting, each with the odds of the opponent winning at the end of that round. If a chunk with the same slot values already exists in declarative memory then it will be merged with the new chunk so that its reference count increases. This way a chunk's activation will reflect the likelihood that it is relevant to a given situation.

After a number of hands have occurred and the model has stored the corresponding chunks in declarative memory, it is able to perform a retrieval to make a prediction on the opponent's hand strength. It does this by specifying the situational parameters of the chunk:

```
+retrieval>
        isa s1hand
        preflop-bets "cc"
```

```
postflop-bets ""
turn-bets ""
river-bets ""
```

The above retrieval request is asking declarative memory for the chunk which best reflects the opponent's hand after a single round of betting consisting of check-check.

Once a chunk is retrieved it can be accessed in the retrieval buffer:

```
=retrieval>
opponentodds =odds
```

It is also possible that no chunk will be retrieved. This will happen if no chunk matches the requested chunk or if none of the matching chunks have an activation higher than a specified threshold. This corresponds to the model being unsure of what the opponent's hand strength is.

Experimental Design

Heads-up matches (1v1) are used to test our agents. Here two agents compete in a series of hands of Texas Hold'em. Each player's winnings are tallied over time and in the end the player with the most money is considered the winner. Typical matches are a few thousand hands. Both player's begin with zero chips, but have unlimited resources. This means that a player may obtain an arbitrarily large negative bankroll. Each player may track their current bankroll in order to allow for changes in strategy based on current trends. Player's are also given standard information on their opponent, i.e. if they call then hands will be revealed. After a match analysis is done based on both the final bankrolls and play history over the course of the match.

For an environment like poker with such variance in style and room for deviation and deception it is very difficult to base a cognitive model solely on human data. Such data would vary drastically between subjects possibly with no particular correlations or consistency. Additionally, it is difficult to gather a significant amount of data on modest human play without deferring to various outside sources. To workaround this difficulty, ACT-R, a cognitive architecture, was used. For our purposes a cognitive architecture is considered to be a set of rules and procedures meant to closely imitate those of the mind at a modest level of abstraction. Statistical techniques consistent with cognition are also used to augment these methods making them more flexible and adaptive. This is then implemented on a computer as a framework which models can be developed on. Ours is precisely such a model currently making use primarily of the statistical techniques for handling the declarative information.

The baseline agent we used to test our opponent models was not particularly complicated. It would first calculate the odds of winning based on the available cards by playing out the hand a number of times using random cards for those unknown. By tallying the number of simulated wins and dividing by the total number of simulations it could get a reasonable estimate of its likelihood of winning. This probability was then compared to an aggression factor (by default 1/2). If the agent's odds of winning were greater than the aggression factor than it would raise, otherwise it would call. This allowed for an easy way to control the agent's level of aggressiveness as the higher the aggression factor the more conservative (a higher likelihood of winning would be required for the agent to raise) it would play. If the agent's opponent raised and based on the previous test it did not wish to re-raise, it would use the pot odds to determine whether to call or fold. Of note is that this agent's betting is directly related to the strength of its hand with better hands being correlated with more aggressive betting. It also does not take the opponent's behavior into consideration.

In response to this baseline agent a few standard opponent modeling techniques were

developed. The fundamental assumption of each is that opponent's future behavior is directly related to recent past behavior. While this is not necessarily the case (players may choose to change strategy in response to their opponent or even independently of them) it was a good starting point clearly capable of modeling our baseline agent. To implement this approach after each hand in which both players reveal their cards (hands ending with a player folding are currently not considered) information pertaining to what the opponent knew at the beginning of each round is saved in memory. This information consists of four chunks (one for each round) each pairing the sequence of betting up to that point with the perceived odds of the opponent winning. Overtime chunks will be replicated resulting in the reinforcement of more common situations. When the model wishes to predict the odds of the opponent winning they can then retrieve the chunk which best represents the current sequence of betting and has the greatest activation. Theoretically, the predicted perceived likelihood of the opponent winning stored in this chunk will approximate the opponent's actual perceived likelihood.

Opponent Models

There are primarily two ways of using this technique to predict opponent's behavior. The first is a specific opponent model (SOM). This model begins a match with no information stored in memory so the only information it has at any given point of play is that which it has accumulated up to that point. The advantage of this model is that it is more tailored to its current opponent (hence specific) and especially to their recent trends in behavior. The latter is true since chunks stored more recently in memory are emphasized more than chunks stored longer ago. This naturally results in recent opponent behavior having a greater impact on the model, which is what we would expect from normal human play. This approach also results in an initially high variance in modeling precision which should become more focused as play goes on. This homing in on the opponent's true strategy is due to the reinforcement of chunks more consistent with the opponent's true style of play and the diminishing strength of outliers, again this behavior naturally emerges and is comparable to our intuition. A key drawback to this type of model though is that it initially has no information on the opponent forcing it to play in the dark. Additionally, it can very easily be deceived by a sufficiently complex opponent aware of its basic technique. This can be accomplished by sudden changes from conservative to aggressive play or vice-versa which will result in the model having to take time to readjust.

The second method addresses our last two concerns about the SOM. It is a general opponent model (GOM). This model begins a match fully equipped with information on a variety of opponents which together form one 'average' opponent. This is analogous to making a prediction based on what would be considered the typical or most common strategy. The model does not store additional information over time and so remains fixed in its initial position. The theoretical advantages of this model are relatively clear. It is more difficult to deceive since the opponent has no way of influencing the information it's basing its predictions on, making their only hope to play in a manner contrary to what is expected. But what is expected is an average of common strategies, each of which is presumably reliable, making opposite behavior absurd. In reality this may not be the case since the model is dependent on what strategies it has been trained against and the strong possibility that opposing strategies may essentially cancel each other out. For instance if you combine an aggressive and conservative player you are left with a neutral or normative player. An additional advantage is that the model begins the match with all its information and does not depend on the opponent to create its model. The significant disadvantage to this approach is that it will most likely not be tailored to the current opponent, reducing its usefulness.

A third approach is a hybrid of the two, combining the specific opponent model and the general opponent model (S+GOM). This model begins the match with information from previous matches, but also accumulates new information during the current match. The combination of the two techniques will hopefully lead to a best-of-both-worlds scenario, or at least a flexible compromise, in which the adaptability of the specific model is combined with the base knowledge of the general.

Results & Discussion

So far all simulations have involved the standard agent competing against an agent making use of either the general or specific opponent model, though data has only been collected for the specific opponent model. The model provides a small advantage over the baseline agent and quickly learns to predict the strength of its opponent's hand. Below is a table of statistics for the comparison of actual and predicted opponent hand strength. The mean and variance for the actual and predicted opponent hand strength is shown for each round. Below that the mean-squared-error for each round is shown.

Preflop	**Actual**	**Predicted**
Mean	0.5	0.5
Variance	0.01	0.0
Flop		
Mean	0.5	0.5
Variance	0.02	0.0
Turn		
Mean	0.5	0.49
Variance	0.04	0.01
River		
Mean	0.49	0.49
Variance	0.07	0.02

Round	**MSE**
preflop	0.01
flop	0.02
turn	0.03
river	0.06

Since hand strength is measured as the probability of winning the hand, it is expected that the mean hand strength will always be 0.5. These are heads-up matches so the average chances of winning should be ½. The variance for the actual hand strength increases for each round as the potential range of hands increases. Even if one has a pair of Aces preflop, the best possible hand at this point, there is still a significant chance that they will not have the best hand after the next three rounds. If one has the best hand at the end of the river, than it's guaranteed that you will win. Likewise, if one has the worst

possible hand at the river, it's guaranteed that you will lose.

The mean-squared error for the model's prediction at each round of play was made based on the statistical concept of an estimator. The ground truth is the opponent's actual hand strength and the estimator is the prediction made by the model. The mean-squared-error was then computed using the following formula:

$$\frac{\sum_{i=1}^{n}(x_i - o_i)^2}{n}$$

Where x_i is the actual opponent hand strength for hand i, o_i is the predicted hand strength for hand i and n is the number of hands.

The increase in mean-squared-error from the preflop to the river reflects the increase in room for error as more cards are dealt on the table. At the preflop the best and worst possible hands are a pair of aces and two-seven off-suit. The former has an approximate probability of winning of 85% while the latter has a 35% chance of winning. As more cards are dealt the range of possible hand strengths will increase until on the river it will reach a maximum of 100% and a minimum of 0% reflecting a guaranteed win or loss. This explains why he model's MSE at each round is approximately the difference in variance between the actual and predicted hand strength.

The prediction is drawn from a more constrained distribution than the actual hand strength. Here the most aggressive betting sequence consists of all raises [crc, rc, rc, rc] and the most conservative all calls [cc, cc, cc, cc]. For our purposes a player's behavior for each round can be considered either aggressive (raise) or passive (c). It is also the case that our baseline bot will tend to play the same way throughout the hand. For instance, if it is dealt a good hand it will raise every round of that hand. This greatly limits the amount of meaningful information that can be obtained from a betting sequence since it is essentially a linear measure of how aggressive the player behaved. This explains why the predicted hand strength has a lower variance than the actual.

In limit poker information on an opponent's hand strength provides only a limited advantage. This is because there is only one possible and small amount one can raise at any point, giving the player a total of three possible actions: fold, call, raise. Consequently, even if it is known to a player that their hand is superior to their opponent's it is difficult to take advantage of the situation. If the player raises then their opponent is likely to fold (since their hand is presumably weak), but if they call they sacrifice their advantage. Theoretically, the additional information from the model should be useful if the opponent has a strong hand, this will allow the player to fold rather than investing himself more in the hand. Specific statistics on this have not yet been gathered.

Future Work

Future work should include expanding the task from limit to no-limit hold'em. This will allow the players to raise more than one possible amount giving the agent's with information on their opponent more leverage, making opponent modeling more important. It also shifts the focus of play from the statistical to the psychological making deception much more important. This is the most important and challenging endeavor that needs to be done.

Players should also take bankroll information into account. Rather than giving each player an infinite bankroll, both players could be given an initial bankroll and then play until one goes bankrupt. This more closely resembles tournament play.

The model should also be tested against a wider range of opponents. It is currently being tested

against a baseline bot primarily making use of pot-odds to decide on an action. The bot does have a couple parameters, such as aggressiveness, which should be varied to increase the utility of the model. Again, the more varied the opponents the more useful modeling information is.

Different representations of past hands should be explored. ACT-R has a retrieval mechanism called partial matching, which allows the model to retrieve a chunk which closely matches the one requested, as opposed to requiring an perfect match. This increases the learning speed of the model since generalizations from related experiences are possible. Blending is also possible. Here instead of retrieving the chunk from memory which matches the one requested and has the highest activation, an average is taken of all those that do match. This could also improve the model.

Alternative modeling techniques can also be explored. So far we have mostly done sub-symbolic modeling, making use of ACT-R's bayesian like retrieval system. Instead of implicitly modeling the opponent by storing past history, the model could replace/augment this with explicitly learning actual parameters reflecting the opponent's play. For instance a numeric value for aggressiveness, randomness/consistency, etc.

Similar to this is to have the model segregate different strategies rather than combining all of them. It can then explicitly learn a prediction for which strategy is currently being used. This would be a more advanced tactic where the model will separate strategies which seem distinct from one another. For instance if it experiences an aggressive player followed by a conservative player, rather than merging the two strategies to get some sort of hedged average it would actually separate the two in memory. It would then be able to model each player more accurately, or even a single player switching between multiple strategies. This would be the technique most similar to what we previously used in the baseball domain.

Conclusion

Our goal for this project was to develop a simple method for modeling an opponent by making inferences about their future actions from their past behavior. In order to emphasize the importance of human cognition over traditional AI, statistical, and machine learning approaches a domain which has the key features of stochasticity, imperfect information, and opportunities for deception was used.

The model developed functions by storing betting sequences with their associated hand strengths. These pairs of information are stored as chunks in ACT-R's long term memory. An opponent's hand strength can then be predicted by making a retrieval request from long term memory asking for the hand strength most associated with a given sequence of betting.

Though our model is able to make reasonable predictions it is currently limited by the lack of variance in play. The more varied the behavior produced by an opponent the better the model can infer the information behind that behavior. This is a limitation of limit hold'em which allows for just a few fixed actions. This makes purely statistical approaches more effective. Future work, regardless of methods, will have to make use of no-limit hold'em, which is known for the importance of taking psychology into consideration. This makes the opponent model far more important and easier to test due to the larger amount of betting options available.

References

Lebiere, C., Gray, R., Salvucci, D., & West, R. (2003). Choice and learning under uncertainty: a case study in baseball batting. In Proceedings of the 25th Annual Conference of the Cognitive Science Society. Mahwah, NJ: Lawrence Erlbaum Associates.

Cohen, A. M., Ritter, F. E., & Haynes, S. R. (2007). Using reflective learning to master opponent strategy in competitive environments. In Proceedings of the International Conference on Cognitive Modeling, 157-162. Oxford, UK: Taylor & Francis/Psychology Press.

D. Billings, D. Papp, J. Schaeffer and D. Szafron, Opponent Modeling in Poker , 1998. AAAI, pp.493-499

Sanner, S., Anderson, J. R., Lebiere, C.,&Lovett, M. C. (2000). Achieving efficient and cognitively plausible learning in backgammon. Paper presented at the 17th International Conference on Machine Learning (ICML-2000), Stanford, CA.

Gray, R. (2002). "Markov at the Bat": A model of cognitively processing in baseball batters. Psychological Science, 13, 542-547.

Anderson, J. R. (1993). ***Rules of*** the ***mind.*** Hillsdale, NJ: Lawrence Erlbaum Associates, Inc

Cohen, A. M., Ritter, F. E., & Haynes, S. R. (2007). Using reflective learning to master opponent strategy in competitive environments. In *Proceedings of the International Conference on Cognitive Modeling,* 157-162. Oxford, UK: Taylor & Francis/Psychology Press.

Sander, J., Lebiere, C. (2008). Hypothesis selection in opponent modeling: a comparative approach. (unpublished)

Mean squared error. (2009, April 22). In *Wikipedia, The Free Encyclopedia*. Retrieved 04:06, April 23, 2009, from http://en.wikipedia.org/w/index.php?title=Mean_squared_error&oldid=285352968