Improved Opponent Modeling in Poker

Aaron Davidson, Darse Billings, Jonathan Schaeffer, Duane Szafron
Department of Computing Science, University of Alberta
Edmonton, Alberta
Canada T6G 2H1

Abstract ***The game of poker has many properties that make it an interesting topic for artificial intelligence (AI). It is a game of imperfect information, which relates to one of the most fundamental problems in computer science: how to handle knowledge that may be erroneous or incomplete. Poker is also one of the few games to be studied where deriving an accurate understanding of each opponent's style is an essential element to success. In developing a strong poker program, the opponent modeling method has always been a central component of the system. As other aspects of the program were improved, the techniques for modeling once again became a limiting factor to the overall level of play. As a result, the topic has been revisited. This paper reports on recent progress achieved by improved statistical methods, which were suggested by experiments using artificial neural networks.***

Keywords: computer poker, imperfect information, opponent modeling

1 Introduction

Poker is a challenging domain, and the goal of producing a strong computer poker player presents many obstacles that have not been faced by other high-performance AI systems. In particular, poker involves hidden information (the opponent's cards), many players (typically nine opponents), deception (bluffing, and handling an opponent's possible bluff), and agent modeling.

The last point is the focus of this paper. To maximize practical results, it is essential to model every opponent, whether they be weak or strong. In most other games, the particular style of the opponent is not very important, because weak moves can be exploited without changing strategy. For example, in chess we can strive to make the objectively best move possible, and simply ignore how the opponent will handle any particular position. In effect, we assume that the opponent plays perfectly (or more precisely, plays at least as well as we do), and this assumption does not adversely affect our choice of "best" move.

In contrast, the method for exploiting weak play in poker entirely depends on the type of mistakes each opponent tends to make. Even among very strong players, there is a wide variety of good styles, and handling each opponent appropriately is a basic requirement for an elite player. The best players are also proficient at adapting to the specific conditions of a game, which may change rapidly over time.

Poker has an underlying mathematical structure, and in theory an optimal strategy exists for playing against perfect opponents. However, determining such a strategy for real poker appears to be computationally infeasible [1]. Furthermore, an optimal strategy would not *maximize* winnings against most typical opponents. Consequently, the issue of opponent modeling cannot be ignored, and is critical to achieving the highest level of play.[1]

[1] This point is not a given. For example, Scrabble^{TM} is technically a game of imperfect information; but this does not play a large role in the overall strategy, and has not prevented the development of a program that is apparently stronger than all human players.

2 Texas Hold'em

The variation of poker examined in this research is Texas Hold'em. This is the most popular form of poker played in North American card clubs and casinos. Hold'em is generally regarded to be the most strategically complex variant of poker commonly played, and is the game of choice for determining the world champion. Despite the richness in strategy, the game is logistically quite simple.

In "10-20 limit Hold'em", each player is dealt two private *hole cards*, face down. The first two players behind the dealer must post blind bets of $5 and $10, respectively, and each player in turn must either *fold*, *call* the current bet, or *raise* an additional $10. On the completion of that first betting round, three community cards, collectively known as the *flop*, are dealt face up on the table, and another betting round ensues, where all bets and raises are exactly $10 (one small bet). Another card, called the *turn*, is dealt face up, followed by a betting round where all bets and raises are doubled to $20 (one large bet). A final *river* card is dealt face up, followed by a final betting round at $20. If more than one player is still active (ie. has not folded), the hole cards are exposed and the winner is determined by the best five-card poker hand, using any combination of the two private cards and the five community cards. A thorough introduction to the strategy of Texas Hold'em can be found in [2].

Our original poker-playing program, *Loki* [3, 4], has been rewritten and is now called *Poki*. The program design, including evaluation of *hand strength*, *draw potential*, *betting strategy*, and "search" by *simulation*, are beyond the scope of this paper [3, 4].

3 Previous Opponent Modeling System

During the course of each hand, a *weight table* is maintained for each opponent. For each possible combination of hole cards, the table gives the probability that the opponent would have played that hand to the present point in the game. Since there are only 1326 two-card combinations, it is convenient to store a value in the range 0.0 to 1.0 for each particular hand. This probability distribution is updated after each opponent action, to be consistent with the betting decisions observed throughout the current hand. The precise details of this *re-weighting* process depends on our method of modeling each opponent.

If we do no opponent modeling at all, we effectively do not update the weight table. All values are fixed, and the probability density function is flat. This is a simple-minded baseline, which ignores all opponent actions.

If we modify the weights as play proceeds, but do it the same way for all players according to some chosen standard for "typical" play, we call it *generic opponent modeling*. For example, we might assume that all opponents will play the same way that we would in each particular situation. This is a vast improvement over no modeling at all, but could be very inaccurate for certain opponents.

Finally, *specific opponent modeling* treats every player as distinct, and utilizes information collected from all previous hands witnessed. While this is obviously preferable to generic opponent modeling in principle, the crude statistical methods used previously were insufficient to show a meaningful advantage. The experiments in this paper involve some fairly straight-forward enhancements to the existing system, which account for more context in the historical record of each player. The result is a significant increase in winning rate, and a clear superiority of the specific techniques over a generic approach.

The basic data structure used for the opponent model is a table of *betting frequencies* for various stages during the hand. The old system consisted of counting the number of times each player folded, called, or raised in each of twelve particular contexts (depending on the betting round (pre-flop, flop, turn, river) and the number of bets to call (zero, one, two or more)).

After each opponent action, the corresponding betting frequencies were used to determine that player's *threshold*, or median hand strength, for the observed action. This in turn was used to estimate the *a posteriori* probability of each possible holding, given its connection to the community cards.

This framework was rather simplistic, as it did not account for many relevant details, such as number of active opponents, and betting position. For example, betting first into many opponents is clearly very different from betting after a single opponent has checked; but with the previous crude modeling, the actions under these different conditions were merged into one betting context. Nevertheless, a fully adaptive re-weighting system based on this information was able to perform as well as the generic opponent modeling system, which was based on a number of expert-defined default values.

4 Improved Opponent Modeling System

There are many other contextual factors that could potentially affect a player's behavior, such as number of active players, relative betting position, size of the pot, and characteristics of the community board cards (eg. the existence of flush or straight draws). Testing each of these factors and tuning their usage would be labourious, and not particularly interesting from a scientific point of view.

Moreover, this approach would be contrary to the philosophy of developing an autonomous system which decides the best action in any situation entirely on its own. Strategies based on a simple rule-based approach are inherently flawed, resulting in a system that contains serious gaps and biases. We believe the inclusion of explicit human knowledge should be avoided whenever possible, in favour of more computer-oriented methods. Historically, this view has been supported by virtually every major success in high-performance game systems, and in many other areas of AI.

Playing poker at a world-class level will require dynamic learning as play proceeds, and the ability to adapt to the prevailing conditions. As such, we have begun investigating alternative methods of accomplishing these tasks, which potentially offer much greater flexibility than the existing structure.

A preliminary study was conducted using an artificial neural network (ANN) for the specific goal of predicting an opponent's next action, based on a full history of a few hundred previous hands by that player [5]. One advantage of using a neural network is that many different parameters can be provided as input, and they will be weighted to maximize the accuracy of the target output, without external intervention. In this way, we can filter out much of the "noise", and identify those features or patterns that are most relevant to the given set of data. This insight will be useful, even if the technique itself cannot be incorporated into a real-time system.

As a result of the ANN study, two particularly strong features for prediction were identified: previous action, and previous amount to call. These properties were added to the existing opponent modeling system to create new contexts, and the performance of the new system was tested empirically.

5 Experimental Results

Poki plays on an online poker server on the Internet Relay Chat (irc.poker.net). Several different poker channels are available, and each game is administered by a dedicated program, or "bot", which deals the cards and prompts each player in turn for an action. No real money is at stake, but statistics are maintained between sessions. All users are eligible to play in the entry level games, called #holdem1. Players who accumulate a large enough bankroll by winning at this level are permitted to play in a more advanced game, called #holdem2. Although the participants are only playing for pride, the majority of people take the game seriously, so it is usually similar to a game in a casino.

This venue has been an important testbed throughout the development of the program. The empirical data gathered in play against actual human opponents has consistently proven to be more reliable than the results of self-play experiments. While playing the program against other versions of itself is a useful diagnostic tool, the inherent biases ("near-sightedness") of this form of testing make real-world experiments indispensable.

The training data for the neural network was based on log files of actual hands played on the IRC poker server by particular opponents. This data was fed into a standard feed-forward ANN (also known as a multilayer perceptron) with four nodes in the hidden layer, and three output nodes for fold, call, or raise. Nineteen different parameters were provided as input nodes, including all of the properties mentioned previously. The back propagation algorithm for neural networks (effectively a local hill-climbing method) was used on repeated iterations of the training data to maximize the prediction of all post-flop betting decisions by that player.

The results of these off-line computations were very encouraging. The actions of real opponents (on independent test data) could routinely be predicted with 80% accuracy, and up to 90% in some cases.

Table 1 demonstrates the accuracy of a typical network with a so-called "confusion matrix" [5]. The columns indicate the predicted frequencies of fold, call and raise, and the rows give the actual frequencies. Values on the main diagonal are correct predictions. For example, 3.3% of the time, the neural net predicted that an opponent would raise when they actually called.

Knowing the *type* of error the network is prone to make is also useful information, because not all errors are equally serious. For example, incorrectly predicting that an opponent will fold can result in a significant error in the calculation of expected value. This was the source of some erratic behavior in previous betting strategies based on run-time simulations. As we can see from the confusion matrix, this type of error is negligible for this particular network and opponent.

		Prediction			
		fold	call	raise	%
Actual	fold	13.0	0.3	0.3	13.6
	call	0.0	58.4	3.3	61.8
	raise	0.0	10.5	14.1	24.7
	%	13.0	69.3	17.7	85.6

Table 1: Neural Net Prediction Accuracy

Figure 1 is an illustration of a neural net predicting the opponent's next action in a particular context. The black area within a node represents the internal value (solid black is 100%), and the thickness of a line corresponds to the strength of that particular signal. Black lines represent a positive correlation, whereas grey lines indicate an inverse relationship. Input 12 is the previous bets to call, while input 11 is the previous action (check/call or bet/raise).

We compared the ANN results to the previous opponent modeling system directly, by using the old system to make the same kind of predictions on the given test data. However, the ANN is an off-line technique, which may or may not eventually be feasible in real time. The results can be used to indicate which input conditions have the greatest influence on the prediction. Two of the strongest factors that were not in the previous opponent modeling system were the opponent's previous action and the previous amount to call. These were used to enhance the existing framework.

Table 2 compares the predictions of the three models on seven different players, ranging in ability from rather weak to fairly strong. The table gives the number of training examples, test examples, prediction rates for the previous, enhanced, and neural net models (in percent), and the strength of the opponent (their overall win rate). The ANN was able to predict the opponent's next action much more reliably, about 81% of the time compared to 57% for the old system.

The program could benefit from a complete re-design of the opponent modeling system,

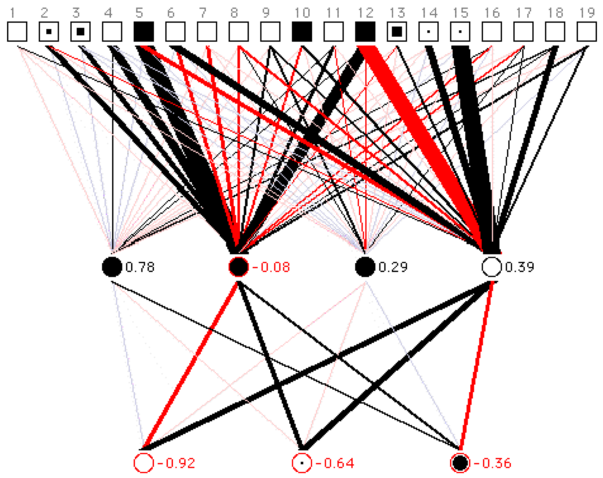

Figure 1: A network after being trained on a specific opponent (predicting a raise)

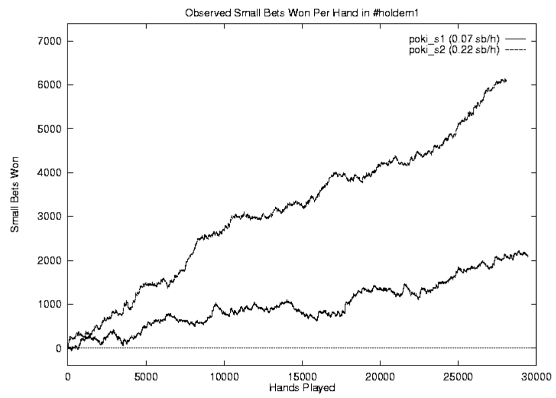

Figure 2: Performance of *Poki* with old and new opponent modeling systems on #holdem1.

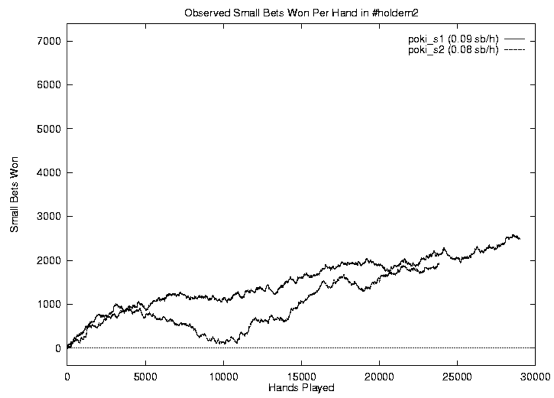

Figure 3: Performance of *Poki* with old and new opponent modeling systems on #holdem2.

which is planned for the near future. However, the results of the hypothetical model suggest that much of this improvement could be realized immediately with fairly simple enhancements to the existing system, having identified the most significant factors.

In order to test this claim, new versions of *Poki* were run on the online poker server, using the refined modeling system with no prior opponent information. The #holdem1 results are shown in Figure 2 and #holdem2 in Figure 3.

To obtain a statistically significant sample, each version must be tested over several thousand hands. The variance in poker is very high, and lengthy runs of good or bad luck are possible. Although most results are fairly stable, anomalies are occasionally observed (one such instance is described below). Common practice is to have each version play at least 20,000 hands at the given level. In order to compare results between games at different levels, the win rate is measured in small bets per hand (sb/h). As a point of reference, an average professional poker player earns in the range of 0.05 to 0.10 sb/h (albeit in much tougher games!).

As a baseline, a version which used no oppo-

Train	Test	Prev	Enhc	ANN	sb/h
218	361	63.4	69.5	90.0	-0.017
250	217	52.1	64.1	75.6	0.131
1323	615	58.2	72.2	80.0	-0.076
237	116	56.0	72.4	75.6	-0.078
325	109	55.1	73.4	82.6	0.127
90	322	51.2	70.2	82.6	0.166
86	138	65.2	80.4	81.2	-0.138
361	268	57.3	71.7	81.1	0.016

Table 2: Comparison of three prediction techniques.

nent modeling whatsoever was tested on #holdem1. This program was unable to win consistently, with a long term average near zero (break-even). In the advanced game, it would have lost quickly.

Poki with the old specific opponent modeling system (poki_s1) won at a rate of approximately +0.09 sb/h in both #holdem1 and #holdem2 games. Full length runs for the generic opponent modeling system were also conducted, resulting in a win rate of approximately +0.08 sb/h for #holdem1, and +0.05 sb/h for #holdem2. This is consistent with our previously reported results, where the two methods had roughly comparable win rates [3].

With the enhanced model (poki_s2), the results against players on #holdem1 improved significantly, to +0.22 sb/h. In contrast, the difference in performance did not appear to be significant for the #holdem2 game, reaching about +0.08 sb/h. However, an anomaly appears to have occurred over a span of 6,000 hands near the beginning of this run.[2] If this negative stretch was indeed primarily due to bad luck, then a better estimate of the final win rate would be at least +0.12 sb/h.

The difference between the results for the #holdem1 and #holdem2 games is interesting. After analyzing the hand evaluations made by *Poki*, it was clear that the new opponent modeling was more committal. Actions of the opponent were given a lot of credit, whether passive or aggressive. This makes it more successful against predictable players, but also more easily deceived against tricky opponents. For example, the program became more vulnerable to a "slowplay", where the opponent does not raise a very strong hand until a later betting round. Since strong players are able to detect this difference over time, they are able to adapt their play to exploit this characteristic.[3] The lesson is that the modeling technique itself should be adaptive, based on the predictability of the opponent.

6 Conclusions and Future Work

In this paper, we revisited the problem of opponent modeling, which is central to the playing ability of a computer poker player. A recurring theme of the research is that improving the program is not a simple linear task, but is a complex system of trade-offs, involving every component of the program. The task of predicting an opponent's next action, based on a large set of contextual information, was investigated with artificial neural networks. The results of these experiments suggested simple but effective changes that could be made to the real-time system.

The modifications to the old modeling system were not extensive, but it is instructive to observe the significant improvements achieved with fairly simple enhancements. Furthermore, it is much more satisfying (and less work!) to have identified these properties with an au-

[2] At the time of this decline, the log of 10,000 commented hands (played over a two week period) was scrutinized. While several distinguishable features of the new modeling system were apparent, there was no obvious explanation for the losing streak, other than having an inordinately large number of good hands lose due to bad luck. Over the following week, the fortune of the program reversed again, and it recovered all of the previous losses.

[3] Other indications of adaptation by the regular players are also evident. In many runs, a noticeable drop in win rate occurs after about 5,000 hands. This curve was less prevalent in versions of *Poki* based on run-time simulations, presumably because the resulting style was less predictable than the conventional approach.

tomated learning system, rather than relying on the input of a human expert. While the domain-specific knowledge of experts may be definitive, it is also notoriously difficult to encorporate and maintain in a high-performance game system.

The topic is far from being well-solved, and we still believe that a thorough re-design of the opponent modeling system is in order. For example, the program still does not make effective use of the information indicated from a showdown. Once the opponent's cards are known, a lot can be inferred from the decisions made during the hand. This can have a significant impact on our understanding of that player's approach to the game, and provide better predictions of future behavior. While other aspects of poker algorithms may eventually approach perfection, this strategic property of the game will likely continue to be a major challenge long into the future.

7 Acknowledgments

Financial support was provided by the Natural Sciences and Engineering Research Council of Canada.

References

[1] D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, 94(1):167–215, 1997.

[2] D. Sklansky and M. Malmuth. *Hold'em Poker for Advanced Players.* Two Plus Two Publishing, 1994.

[3] D. Billings, D. Papp, J. Schaeffer, and D. Szafron. Opponent modeling in poker. In *AAAI National Conference*, pages 493–499, 1998.

[4] D. Billings, L. Pena, J. Schaeffer, and D. Szafron. Using probabilistic knowledge and simulation to play poker. In *AAAI National Conference*, pages 697–703, 1999.

[5] A. Davidson. Using artificial neural networks to model opponents in Texas Hold'em, 1999. www.cs.ualberta.ca/~davidson/poker/nnpoker.pdf.