Implicit Opponent Modelling via Dynamic Case-Base Selection

Jonathan Rubin and Ian Watson

Department of Computer Science
University of Auckland Game AI Group, New Zealand
jrub001@aucklanduni.ac.nz, ian@cs.auckland.ac.nz
http://www.cs.auckland.ac.nz/research/gameai

Abstract. In this paper we introduce and evaluate an approach for performing implicit opponent modelling by constructing multiple, distinct case-bases via expert imitation and dynamically selecting the best case-base to use against a particular opponent at runtime. In our experimental results, we address whether a strategy based on dynamic case-base selection can improve overall performance against a range of competitors, compared to the use of a single case-base strategy. We apply this approach to the stochastic, imperfect information domain of both limit and no limit two-player Texas Hold'em poker.

1 Introduction

In adversarial environments, any agent that considers information about an opponent and uses this to adapt its own strategy is said to perform **opponent modelling**. Successful opponent modelling has the ability to dramatically improve a system's performance [1]. Opponent modelling can be either **explicit** or **implicit**. *Explicit opponent modelling* attempts to learn a set of parameters associated with an opponent's strategy via a series of observations. Once the parameters are known, the agent adjusts their own strategy, in order to take advantage of the opponent. On the other hand, *implicit opponent modelling* does not directly attempt to decipher the details of an opponent's strategy, but instead selects an appropriate response from a set of strategies, based on the performance of those strategies against the opponent. In this paper, we introduce and evaluate an implicit opponent modelling approach, based on the dynamic selection of separate case-bases constructed via *expert imitation*.

Our research takes place in the domain of two-player Texas Hold'em poker – a game defined by simple rules, which however, provides a rich, dynamic environment for applying sophisticated strategies. Performance in the game of poker is known to have an intransitive relationship, i.e. while player A beats player B and player B beats player C, it does not necessarily follow that A beats C. Given the intransitive nature of poker strategies, the successful combination of different styles of play has the potential to largely improve overall performance, against a range of different opponents.

By training on experts with different styles of play, we create multiple case-bases in the domains of limit and no limit Texas Hold'em. We evaluate what impact implicit opponent modelling via dynamic case-base selection has on overall performance. Specifically, we address whether dynamic case-base selection can improve performance, compared to use of a single case-base strategy.

2 Related Work

The *lazy learning* [2] of case-based reasoning is particularly well suited to expert imitation where expert observations can be recorded and stored for later use at decision time. In the games domain, [3] demonstrated successful imitation in the domain of simulated robot soccer to control a multi-agent soccer team. Ontañón et al. [4] observes human experts playing a real time strategy game and uses case-based planning to generate unique strategies. Our own case-based poker playing system, Sartre [5], has demonstrated strong performance at international computer poker competitions by training on hand history data from the strongest agent in the previous year's competition.

The works discussed so far use *expert imitation* to construct a single case-base. Here, we use *expert imitation* to construct multiple case-bases. Each case-base is constructed by processing a subset of data that contains the playing decisions of a single expert. Using this approach we are easily able to derive a set of different playing styles. Given a set of distinct case-bases, we then attempt to dynamically select the case-base that achieves the greatest profit against a particular opponent in order to maximise overall profit against a range of players.

A similar approach was originally applied to the game of limit Texas Holdem in [6, 7] where the UCB1 [8] allocation strategy was applied to dynamically select experts during play. Our procedure for dynamic selection from multiple case-bases, is an adapted version of that described in [7]. Our own research extends this idea to the more complicated domain of no limit Texas Holdem. Furthermore, our work focuses specifically on implicit opponent modelling via dynamic case-base selection, where individual case-bases have been constructed via expert imitation. A major benefit of using this approach is that producing new strategies simply requires updating the data that is used to train the system. Using this approach a diverse set of strategies with various playing styles can be produced with negligible computational effort.

3 Dynamic Case-Base Selection

A case-base is made up of a collection of cases, $C_j = \{c_{j,1}, c_{j,2}, \ldots, c_{j,n}\}$, where j refers to an individual case-base and $\forall c \in C_j, c = (x, a)$, where x is a feature vector that captures game state information and a is an action vector that specifies the probability of taking a certain action, given game state x. The following features describe the current state of the game: 1) Hand strength, 2) Previous betting history, 3) Public card information and 4) Stack commitment (no limit only). A more detailed discussion of the case representation can be found in [5].

Given a game state described by c_t, a decision is made by processing the collection of stored cases and maximising a global similarity metric to retrieve the most similar case to c_t.

$$c_{max} = \arg\max_{c_k} sim(c_k, c_t), \forall c_k \in C_j \tag{1}$$

where, $sim(c_1, c_2)$ is some global similarity metric that determines the similarity between the feature vectors of the two cases c_1, c_2.

3.1 Action Vectors

The action vectors used to make a playing decision differ depending on the domain.

Limit Hold'em In the domain of limit Hold'em the action vector used by the expert imitator has the following format:

$$a = (f, c, r)$$

where each entry within the vector is a number between 0.0 and 1.0 that indicates the probability of taking a particular betting action. All entries within the vector must sum to exactly 1.0. The actions that are possible in limit Hold'em are either fold (f), check/call (c) or bet/raise (r).

No Limit Hold'em In limit Hold'em, if a player wishes to raise, the amount they are allowed to raise is set at a certain predefined level. On the other hand, in the no limit variation, players are allowed to bet or raise any amount, including all of the money or chips they possess (this is called going *all-in*). While this appears to be a simple rule change, the consequences it has on game play are quite dramatic. In order to construct an action vector for no limit Hold'em, a mapping is required to assign quantitative bet amounts into discrete categories. The no limit action vector used by the expert imitators in this paper is given by the following:

$$a = (f, c, q, h, i, p, d, v, t, a)$$

where each entry refers to different discrete betting categories as follows: fold (f), call (c), raise quarter pot (q), raise half pot (h), raise three quarter pot (i), raise pot (p), raise double pot (d), raise five times pot (v), raise ten times pot (t), all-in (a). Once again each entry corresponds to the probability of taking that particular action and all entries in the vector sum to 1.0.

3.2 UCB1

Given a collection of case-bases, $\{C_j : j = 1 \dots N\}$, we perform implicit opponent modelling by dynamically selecting an appropriate case-base to use against a particular opponent. An individual case-base is dynamically selected at the beginning of every hand. In order to dynamically select an appropriate case-base at runtime, the UCB1 [8] algorithm is used. The UCB1 algorithm offers a simple, computationally efficient solution to the **exploration/exploitation** trade off. Successfully handling the trade off between exploration and exploitation is a problem that is consistently faced in statistics and artificial intelligence. Decisions need to be made as to whether to select a strategy that has so far offered the greatest reward (*exploitation*), versus selecting an alternative strategy that may offer yet a greater reward than that witnessed so far (*exploration*).

The UCB1 algorithm defines a policy for strategy selection based on the concept of **regret**. *Regret* refers to the difference between the reward that would have been received, had the optimal strategy always been selected, compared to the actual reward received. The UCB1 algorithm offers a logarithmic bound on regret. The following application of UCB1 is used to dynamically select from a collection of case-bases:

1. Select each case-base once
2. Select case-base, C_j, that maximises the following equation:

$$\arg\max_j \bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}} \tag{2}$$

where,
$\bar{x}_j$ is the average amount won by case-base C_j, when challenging the current opponent,
n is the total number of hands played so far, and
n_j is the total number of times case-base C_j has been selected.

The UCB1 algorithm above determines the order in which to select different case-bases, given their performance against a particular opponent. In the computer poker domain, $\bar{x}_j$ in equation (2) refers to the average utility (profit or loss) achieved by case-base, C_j, while playing against the current opponent. Similarly, n_j refers to the total number of times C_j has been chosen to play against the current opponent. However, due to the inherent variance present in the game of Texas Hold'em, strategy selection based on profit alone can severely bias the results of the UCB1 allocation policy. Hence, efforts to reduce this inherent variance are required in order to stabilise the results.

3.3 Variance Reduction

As in [7], we employ the use of DIVAT analysis [9] in order to improve the average utility values, $\bar{x}_j$, used by the UCB1 allocation strategy. DIVAT (ignorant value assessment tool) is a perfect information variance reduction tool developed by

the University of Alberta Computer Poker Research Group[1]. The basic idea behind DIVAT is to evaluate a poker hand based on the expected value (EV) of a player's decisions, not the actual outcome of those decisions. DIVAT achieves this by comparing the EV of the player's decisions against the EV of some baseline strategy, see Equation 3.

$$DivatOutcome = EV(ActualActions) - EV(BaselineActions) \quad (3)$$

Equation 3 attempts to factor out the effects of luck as both the actual strategy and the baseline strategy experience the same *lucky* or *unlucky* sequence of events. For example, a strategy that benefits by a statistically unlikely event no longer makes a difference as the baseline strategy also benefits by the same improbable event. What actually matters is any difference in EV the strategy is able to achieve by varying the actions that it takes. When the strategy is able to achieve a greater EV than the baseline, it is rewarded. Alternatively, if the strategy's actions result in a lower EV, then it is punished with a negative outcome. If the EV of both strategies is the same the outcome is 0.

Any type of strategy can be used as a baseline. Typically, in limit Hold'em a bet-for-value baseline strategy is adopted. A bet-for-value strategy makes no attempt to disguise the strength of its hand. By establishing game-theoretic equilibrium-based thresholds, a sequence of baseline betting decisions can be constructed, based on hand strength alone. Strong hands will be bet and/or raised, whereas weak hands will be checked or folded.

Finally, DIVAT requires perfect information about both players' hidden cards. When either player folds this information is not available. Therefore, to calculate the utility values used by the UCB1 allocation strategy, DIVAT analysis is employed only when a showdown occurs, i.e. when both players reveal their hidden cards. If a fold occurs the actual monetary outcome of the hand is used instead.

4 Experimental Results

4.1 Methodology

We provide experimental results in the domains of both limit and no limit two-player Texas Hold'em. In each domain 3 separate case-bases were constructed via expert imitation. Each case-base was constructed by processing a separate subset of hand history data from the Annual Computer Poker Competition [10] as follows:

1. The first case-base was constructed by processing the decisions made by the agent that won the total bankroll division of the 2010 AAAI computer poker competition.

[1] `http://webdocs.cs.ualberta.ca/~games/poker/index.html`

2. The second case-base was constructed by processing the decisions made by the agent that won the instant run-off division of the 2010 AAAI computer poker competition.
3. The final case-base was that used by our own entry into the 2010 AAAI computer poker competition.

A Dynamic strategy was constructed that dynamically selects case-bases at runtime to use against the current opponent. Dynamic case-base selection was based on the UCB1 allocation strategy together with showdown DIVAT analysis. In the limit variation, a bet-for-value baseline strategy was adopted during DIVAT analysis. In no limit a more simplistic *always-call* strategy was used as the baseline.

The Dynamic strategy, along with all three single case-base strategies, were then challenged against two different types of computerised players. All matches played were duplicate matches. A duplicate match involves playing the same 3000 hands twice. In the first run, 3000 unique hands are played, after which the players' memories are wiped and the 3000 hands are played again, but in the reverse direction, i.e. the cards that were initially given to player A are instead given to player B. This way both players get to play both sets of cards and this reduces the variance that is involved in simply playing a set of hands in one direction only. All case-based strategies played 5 duplicate matches against each of the two computerised opponents. In total, 420,000 hands of poker were played.

4.2 Limit Results

Table 1. Results against Fell Omen 2 and AlistairBot

Strategy	**Fell Omen 2**		**Strategy**	**AlistairBot**
Dynamic	0.0134 ±0.006		Dynamic	0.6885 ±0.009
Sartre	-0.0050 ±0.008		Rockhopper	0.6669 ± 0.015
PULPO	-0.0056 ±0.009		PULPO	0.6472 ± 0.021
Rockhopper	-0.0219 ±0.010		Sartre	0.6422 ± 0.017
Strategy			**Average**	
1. Dynamic			0.3510 ± 0.008	
2. Rockhopper			0.3225 ± 0.013	
3. PULPO			0.3208 ± 0.015	
4. Sartre			0.3186 ± 0.014	

In the limit domain each strategy challenged the following competitors:

1. **Fell Omen 2** – A solid Nash equilibrium-based agent commonly used as a benchmark for testing limit Hold'em agents.
2. **AlistairBot** – An exploitive agent that uses Monte-Carlo simulation to determine the decision with the best EV against the current opponent.

Table 1 presents the results of the four strategies against the above two competitors. The results are presented in small bets per hand (sb/h), where the total number of small bets won or lost are divided by the number of hands played. The single case-base strategies are identified by the name of the original expert used to train the system. Rockhopper and PULPO were the first place finishers within the total bankroll and instant run-off divisions of the 2010 two-player, limit competition, respectively. The Sartre agent was our entry into the 2010 computer poker competition.

4.3 No Limit Results

In the no limit domain, each strategy challenged the following opponents:

1. **MCTSBot** – an exploitive agent that uses Monte-Carlo Tree Search [11].
2. **SimpleBot** – a no limit rule-based agent.

The *opentestbed*[2] project was used to gather results as the above no limit agents were made publicly available within this framework. Table 2 presents the no limit results. Hyperborean was the winner of the instant run-off division and Tartanian was the winner of the total bankroll division of the 2010 two-player, no limit competition. Sartre was our own (no limit) entry into the 2010 computer poker competition. The results are in big blinds per hand.

Table 2. Results (No Limit) against MCTSBot and SimpleBot

Strategy	**MCTSBot**		**Strategy**	**SimpleBot**
Hyperborean	1.4409 ± 0.179		Hyperborean	0.6505 ± 0.089
Dynamic	1.3332 ± 0.146		Dynamic	0.5928 ± 0.058
Sartre	0.6919 ± 0.112		Sartre	0.4591 ± 0.053
Tartanian	0.0258 ± 0.289		Tartanian	0.3836 ± 0.060
Strategy			**Average**	
1. Hyperborean			1.0457 ± 0.134	
2. Dynamic			0.9630 ± 0.102	
3. Sartre			0.5755 ± 0.083	
4. Tartanian			0.2047 ± 0.175	

4.4 Discussion

The results presented in Table 1 support the idea that implicit opponent modelling via dynamic case-base selection can improve performance, compared to the use of a single case-base.

[2] http://code.google.com/p/opentestbed/

Given the non-transitive performance relationship that exists in the poker domain, a single unchanging strategy can typically not be expected to perform the best against every opponent it faces. Intuitively, it makes sense that, given a set of case-bases to choose from, if we are able to identify the unique best case-base to use against each individual opponent, overall performance should improve. We expect this result as long as there is enough diversity in our set of case-bases such that there is no single case-base that performs best against all opponents. The results from Table 1 suggest that, in the limit Hold'em experimental domain, there is enough diversity provided by the three case-bases to achieve a beneficial result.

On the other hand, in the more complicated no limit Hold'em environment, a similar improvement in overall performance is not observed. Table 2 shows that while Dynamic provides an improvement over two of the single case-bases, it is not able to achieve a greater performance compared to the Hyperborean case-base. In the no limit domain, it appears that there is not enough diversity in the three case-bases provided to positively affect overall performance. Notice that, in Table 2, the relative ordering of the single case-base strategies does not change in response to challenging different opponents, i.e. the Hyperborean case-base always achieves the best result, followed by Sartre and then Tartantian. Compare this to the results in Table 1, where the relative ordering does change. This suggests that in order to improve overall performance, either a larger, more diverse set of case-bases is required or a further set of opponents (with different playing styles) needs to be challenged, in order to receive any beneficial results.

5 Conclusion

In conclusion, we have presented an approach for performing implicit opponent modelling by constructing multiple, distinct case-bases via expert imitation and dynamically selecting the best case-base to use against a particular opponent at runtime. We applied the approach to the stochastic, imperfect information domain of limit and no limit Texas Hold'em poker. While the results obtained were somewhat mixed, the approach does appear to have promise. It is the opinion of the authors that the outcome of the experiments performed in the no limit Hold'em environment suggest that a more diverse set of case-bases are required before further benefits can be obtained over the use of a single case-base. Furthermore, while dynamic case-base selection was not able to achieve the greatest overall profit (in this domain), it still managed to outperform two of the single case-base strategies. On the other hand, results from the limit Hold'em domain support the intuition that, given a set of case-bases to choose from, identifying the best case-base to use against each individual opponent, has the ability to improve overall performance compared to use of a single case-base alone.

References

1. Kennard Laviers, Gita Sukthankar, David W. Aha, and Matthew Molineaux. Improving Offensive Performance Through Opponent Modeling. In *Proceedings of the Fifth Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2009*, 2009.
2. David W. Aha. Editorial. *Artificial Intelligence Review*, 11(1-5):7–10, 1997.
3. Michael W. Floyd and Babak Esfandiari. An Active Approach to Automatic Case Generation. In *Case-Based Reasoning Research and Development, 8th International Conference on Case-Based Reasoning, ICCBR 2009*, pages 150–164, 2009.
4. Santiago Ontañón, Kane Bonnette, Prafulla Mahindrakar, Marco A. Gómez-Martín, Katie Long, Jainarayan Radhakrishnan, Rushabh Shah, and Ashwin Ram. Learning from Human Demonstrations for Real-Time Case-Based Planning. In *IJCAI-09 Workshop on Learning Structural Knowledge From Observations*, 2009.
5. Jonathan Rubin and Ian Watson. Similarity-Based Retrieval and Solution Re-use Policies in the Game of Texas Hold'em. In *Case-Based Reasoning. Research and Development, 18th International Conference on Case-Based Reasoning, ICCBR 2010*, pages 465–479, 2010.
6. Michael Johanson, Martin Zinkevich, and Michael H. Bowling. Computing Robust Counter-Strategies. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, 2007.
7. Michael Bradley Johanson. Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player. Master's thesis, University of Alberta, 2007.
8. Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256, 2002.
9. Darse Billings and Morgan Kan. A Tool for the Direct Assessment of Poker Decisions. *The International Association of Computer Games Journal*, 2006.
10. University of Alberta CPRG. The Annual Computer Poker Competition, 2010. http://www.computerpokercompetition.org/.
11. Guy Van den Broeck, Kurt Driessens, and Jan Ramon. Monte-Carlo Tree Search in Poker Using Expected Reward Distributions. In *Advances in Machine Learning, First Asian Conference on Machine Learning, ACML 2009*, pages 367–381, 2009.