Learning a Value Analysis Tool For Agent Evaluation

Martha White
Department of Computing Science
University of Alberta
whitem@cs.ualberta.ca

Michael Bowling
Department of Computing Science
University of Alberta
bowling@cs.ualberta.ca

Abstract

Evaluating an agent's performance in a stochastic setting is necessary for agent development, scientific evaluation, and competitions. Traditionally, evaluation is done using Monte Carlo estimation; the magnitude of the stochasticity in the domain or the high cost of sampling, however, can often prevent the approach from resulting in statistically significant conclusions. Recently, an advantage sum technique has been proposed for constructing unbiased, low variance estimates of agent performance. The technique requires an expert to define a value function over states of the system, essentially a guess of the state's unknown value. In this work, we propose learning this value function from past interactions between agents in some target population. Our learned value functions have two key advantages: they can be applied in domains where no expert value function is available and they can result in tuned evaluation for a specific population of agents (e.g., novice versus advanced agents). We demonstrate these two advantages in the domain of poker. We show that we can reduce variance over state-of-the-art estimators for a specific population of limit poker players as well as construct the first variance reducing estimators for no-limit poker and multi-player limit poker.

1 Introduction

Evaluating an agent's performance is a common task. It is a critical step in any agent development cycle, necessary for scientific evaluation and useful for determining the results of a competition, which have become popular in the artificial intelligence community. In some cases it is possible to compute an exact measure of performance, but only for small or deterministic domains. In the other cases, the most common approach is to use Monte Carlo estimation. The agent completes a number of repeated independent trials of interaction in the domain, and a sample average of its performance is used as an estimate. The magnitude of the stochasticity in the domain, however, may prevent the approach from resulting in statistically significant conclusions, particularly if the cost of trials is high.

Recently, Zinkevich and colleagues [2006] proposed an *advantage-sum* technique for constructing a low-variance unbiased estimator for an agent's performance. The estimator examines the complete history of interaction for a trial, unlike Monte Carlo which uses only the single value (utility) of the agent's per-trial performance. As the estimator is provably unbiased (matching the player's realized utility in expectation), a sample average using the estimator provides an alternative, potentially lower-variance, estimate of an agent's performance. Unfortunately, the approach requires a domain-specific value function to be provided, which much satisfy certain constraints. Furthermore, the variance reduction of the resulting estimator depends entirely on the quality of the provided value function. This limits the applicability of the approach to well-understood domains for which a value function can be constructed. Although the approach was successfully used to achieve dramatic variance reduction in evaluating play for two-player, limit Texas hold'em poker [Billings and Kan, 2006], the difficulty in hand-crafting appropriate value functions for other domains is limiting.

In this paper, we propose using machine learning to find a value function to be used in an advantage-sum estimator. We define an optimization to directly minimize the estimator's variance on a set of training data and derive a closed form solution for this optimization in the case of linear value functions. The optimization results in two distinct advantages. First, it can be more easily applied to new domains. Instead of requiring a hand-crafted, domain-specific value function, our approach only requires a set of domain-specific features along with data from previous interactions by similar agents. In fact, if a good hand-crafted value function is already known, it can be provided as a feature, and the optimization can result in further variance reduction. Second, our approach can find an estimator for agent evaluation that is tuned to a specific population of agent behavior (e.g., advanced behavior or novice behavior) by providing such data in training. The approach is general, applying to a wide variety of domains: single agent and multi-agent domains; fully and partially observable settings; as well as zero-sum and general-sum games. We demonstrate the efficacy of the approach on the classic two-player limit poker game and two poker domains for which no previous variance reduction estimator exists: a two-player no-limit and a 6-player limit game.

2 Background

We will first describe extensive games as a general model for (multi-)agent interaction. We will then discuss previous approaches to agent evaluation.

2.1 Extensive Games

Definition 1 *[Osborne and Rubinstein, 1994, p. 200] A finite extensive game with imperfect information has the following components:*

- *A finite set, N, of **players**.*
- *A finite set ,H, of sequences, the possible **histories** of actions, such that the empty sequence is in H and every prefix of a sequence in H is also in H. $Z \subseteq H$ are the **terminal histories** (those which are not a prefix of any other sequences). $A(h) = \{a : ha \in H\}$ are the actions available after a non-terminal history $h \in H$[1]. To denote that a history h is a prefix of a terminal history z, we will write $h \sqsubseteq z$. ,*
- *A **player function** P that assigns to each non-terminal history, $h \in H \backslash Z$, a member of $N \cup \{c\}$, where c represents chance. $P(h)$ is the player who takes an action after the history h. If $P(h) = c$, then chance determines the action taken after history h.*
- *A function f_c on $\{h \in H : P(h) = c\}$ associating to each such history a probability measure $f_c(\cdot|h)$ on $A(h)$ ($f_c(a|h)$ is the probability that a occurs given h). Each probability measure $f_c(\cdot|h)$ for a given h is independent of the other probability measures $f_c(\cdot|h'),\ h' \neq h$.*
- *For each player $i \in N$ a partition $\mathbf{I}_i$ of $\{h \in H : P(h) = i\}$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. $\mathbf{I}_i$ is the **information partition** of player i; a set $I_i \in \mathbf{I}_i$ is an **information set** of player i.*
- *For each player $i \in N$ a utility function u_i from the terminal states Z to the reals $\mathbf{R}$.*

The extensive game formalism is a very general model of sequential decision-making and encapsulates finite-horizon POMDPs (where $|N| = 1$), finite-horizon MDPs (where $|N| = 1$ and $\forall I_i\ |I_i| = 1$), and n-player general-sum or zero-sum games. The only additional assumption made in this work is that the game Γ (but not any player's policy) is known.

A **strategy of player i**, σ_i in an extensive game is a function that assigns a distribution over $A(I_i)$ to each $I_i \in \mathbf{I}_i$. A **strategy profile** σ consists of a strategy for each player, $\sigma = (\sigma_1, \sigma_2, \ldots \sigma_{|N|})$. The goal of agent evaluation is to estimate the expected utility of some player $j \in N$ given a strategy profile, i.e.,

$$U_j = E_z \left[u_j(z) | \sigma \right]. \tag{1}$$

If the extensive game is small, one can compute this expectation exactly by enumerating the terminal histories. In large games, however, this approach is not practical.

[1] We write ha to refer to the sequence with action a concatenated to history h.

2.2 Monte Carlo Estimation

The traditional approach to agent evaluation is to estimate the expectation in Equation (1) by sampling. The agents repeatedly interact with the environment, drawing independent samples $z_1, \ldots, z_T$ from the distribution $\Pr(z|\sigma)$. The estimator is simply the average utility,

$$\hat{U}_j = \frac{1}{T} \sum_t u_j(z_t). \tag{2}$$

This estimator is unbiased (i.e., $E[U_j|\sigma] = E[u_j(z)|\sigma]$), and so the mean-squared-error (MSE) of the estimate is its variance,

$$\mathrm{MSE}(\hat{U}_j) = \mathrm{Var}\left[\hat{U}_j|\sigma\right] = \frac{1}{T}\mathrm{Var}\left[u_j(z)|\sigma\right]. \tag{3}$$

This approach is effective when the domain has little stochasticity (i.e. $\mathrm{Var}\left[u_j(z)\right]$ is small), agent trials are cheap (i.e. T can be made large), and/or the required precision is not small (i.e. large $\mathrm{Var}(\hat{U}_j)$ is tolerable). If trials are expensive relative to the domain's stochasticity and required precision, then it may not be possible to make statistically significant conclusions based on this estimate. This limitation often arises in situations involving human or physical robot participants. For example, in one particular poker game (two-player $1/2$ limit Texas hold'em), the standard deviation of a player's outcome is around \$6 and a typical desired precision is around \$0.05. This precision would require more than fifty thousand trials to achieve. If one or more of the players is a human (or many agent pairings must be evaluated), this large number of trials is impractical.

One approach to improving the Monte Carlo estimator is to find a better estimate of per-trial utility. The goal is to identify a real-valued function on terminal histories $\hat{u}_j(z)$ where,

$$\forall \sigma \quad E_z \left[\hat{u}_j(z) | \sigma\right] = E_z \left[u_j(z)|\sigma\right]. \tag{4}$$

In other words, Equation (4) means that $\hat{u}_j(z)$ is an unbiased estimator of $u_j(z)$. If the variance of $\hat{u}_j(z)$ is lower than the variance of $u_j(z)$, then we can use $\hat{u}_j$ in place of u_j in Equation (2) to get an improved estimator.

Wolfe [Wolfe, 2002] employed this approach in evaluating performance in blackjack. Using a baseline policy with a known expected performance, Wolfe compares the player's winnings on a given hand with the winnings the baseline policy would have attained had it been employed. The resulting unbiased estimator is this difference added to the baseline policy's known expected winnings. Wolfe showed the approach could result in a 50-fold reduction in the standard deviation of the resulting estimator for evaluation in blackjack. The approach, however, is limited to single-agent settings and tends to over penalize near-optimal decisions with unlucky outcomes.

2.3 Advantage Sum Estimators

Recently, Zinkevich and colleagues [2006] introduced a general approach to constructing low-variance estimators for sequential decision-making settings. Assume one is given a real-valued function on histories $V_j : H \rightarrow \Re$. Define the following real-valued functions on terminal histories,

$$S_{V_j}(z) = \sum_{\substack{ha \sqsubseteq z \\ P(h) \neq c}} V_j(ha) - V_j(h) \tag{5}$$

$$L_{V_j}(z) = \sum_{\substack{ha \sqsubseteq z \\ P(h) = c}} V_j(ha) - V_j(h) \tag{6}$$

$$P_{V_J} = V_j(\emptyset). \tag{7}$$

We will call S_{V_j} the **skill** for player j, L_{V_j} the **luck** for player j, and P_{V_j} the value of player j's **position** (a constant). We label these functions skill and luck because the skill is obtained from changes in the value function due to the agents' actions (i.e. its advantages) and the luck from changes in the value function due to actions by chance. Notice that the skill function for player j includes actions for all players, and so all the players' actions affect all players' skill terms.The advantage sum estimator is now simply,

$$\hat{u}_{V_j}(z) = S_{V_j}(z) + P_{V_j}. \tag{8}$$

Using the fact that terms cancel when summing skill, luck, and position, we can see that,

$$u_j(z) = S_{V_j}(z) + L_{V_j}(z) + P_{V_j}. \tag{9}$$

If V_j is chosen carefully so that $E\left[L_{V_j}(z)|\sigma\right]$ is zero (the zero-luck constraint), then $\hat{u}_{V_j}$ is an unbiased estimator. Additionally, the advantage sum estimator subsumes Wolfe's approach for one-player games.

DIVAT, the Ignorant Value Assessment Tool, implements the advantage sum idea for two-player, limit Texas hold'em using a hand-designed value function shown to satisfy the zero-luck constraint. The resulting estimator has proven to be very powerful, with approximately a three-fold reduction in per-trial variance. As a result, nine-times fewer hands are required to draw the same statistical conclusions. Though DIVAT has been successful in two-player, limit poker, the hand-designed value function does not extend to other (poker) games. Moreover, no useful reduced-variance estimator has been identified for no-limit poker or games involving more than two players.

3 Our Approach: MIVAT

A more general approach to using the advantage sum estimator from Equation (5) is to learn a good value function from past interactions between players. This approach has two advantages. First, designing a value function can be difficult, particularly in the case of limited domain knowledge. Second, learning a more specific value function to a group of players (using a population of such players as the training data) can further reduce variance in the assessment for those players. This section defines an optimization for finding the ideal value function, and then derives a closed-form solution for the space of linear value functions. We call this general approach and closed-form solution to the optimization MIVAT, the Informed Value Assessment Tool.

Our goal is to minimize the variance of $\hat{u}_{V_j}(z)$. Using Equation (9), we can rewrite the advantage sum estimator only in terms of the outcome's utility and luck,

$$\hat{u}_{V_j}(z) = u_j(z) - L_{V_j}(z). \tag{10}$$

In order to ensure V_j satisfies $E\left[L_{V_j}(z)|\sigma\right] = 0$ (the zero-luck constraint), let's assume that $V_j(ha)$ is only provided for histories where $P(h) = c$, i.e. on histories directly following a chance node.[2] We then define V_j for the remaining histories, where chance is next to act, to explicitly satisfy the zero-luck constraint,

$$V_j(h \text{ s.t. } P(h) = c) \equiv \sum_{a' \in A(h)} f_c(a'|h) V_j(ha'), \tag{11}$$

so then,

$$L_{V_j}(z) = \sum_{\substack{ha \sqsubseteq z \\ P(h) = c}} \left(V_j(ha) - \sum_{a' \in A(h)} f_c(a'|h) V_j(ha') \right). \tag{12}$$

This reformulation of the advantage-sum estimator has two benefits. First, we need only define a value function for the histories directly following chance nodes. Second, the value function is guaranteed to be unbiased. Because of this guarantee, we are now unconstrained in our choice of value function.

Our goal is to find a value function that minimizes the variance of the advantage-sum estimator. The variance of the estimator depends upon the unknown agent strategies in the target population. We presume that we have samples of outcomes $z_1, \ldots, z_T$ from agents in this population, and use the estimator's sample variance as a proxy for the true variance. This gives us the following target optimization.

$$\underset{V_j}{\textbf{Minimize:}} \quad \sum_{t=1}^{T} \left(\hat{u}_{V_j}(z_t) - \frac{1}{T} \sum_{t'=1}^{T} \hat{u}_{V_j}(z_{t'}) \right)^2$$

3.1 Linear Value Function

In order to make the above optimization tractable, we focus on the case where V_j is a linear function. Let $\phi : H \to \mathbf{R}^d$ map histories to a vector of d features. We require,

$$V_j(h) = \phi(h)^T \theta_j, \tag{13}$$

for some $\theta_j \in \mathbf{R}^d$. We will use $\hat{u}_{\theta_j}$ as shorthand for the advantage-sum estimator from Equation (10), where V_j is defined as above.

We now derive a closed-form solution, θ_j^*, to our optimization. Let $u_t = u_j(z_t)$. From Equations (10) and (12), we get,

$$\hat{u}_{\theta_j}(z_t) = u_t - \left[\sum_{\substack{ha \sqsubseteq z_t \\ P(h) = c}} \left(\phi(ha) - \sum_{a' \in A(h)} f_c(a'|h) \phi(ha') \right) \right]^T \theta_j \tag{14}$$

[2] For simplicity, we assume that if $P(ha) = c$ then $P(h) \neq c$, i.e., the chance player never gets consecutive actions. A trivial modification can always make a game satisfy this constraint.

Define the following shorthand notation,

$$A_t = \sum_{\substack{ha \sqsubseteq z_t \\ P(h)=c}} \left(\phi(ha) - \sum_{a' \in A(h)} f_c(a'|h)\phi(ha') \right) \quad (15)$$

$$A = \frac{1}{T}\sum_{t=1}^{T} A_t \qquad \bar{u} = \frac{1}{T}\sum_{t=1}^{T} u_t \quad (16)$$

Then, $\hat{u}_{\theta_j}(z_t) = u_t - A_t^T\theta_j$ and $\frac{1}{T}\sum_{t'=1}^{T}\hat{u}_{\theta_j}(z_{t'}) = \bar{u} - A^T\theta_j$, and we get the following optimization.

$$\underset{\theta_j \in \mathbf{R}^d}{\textbf{Minimize:}} \; C(\theta_j) = \sum_{t=1}^{T}\left[(u_t - \bar{u}) - (A_t - A)^T\theta_j\right]^2$$

As our objective is convex in θ_j, we can solve this optimization by setting the objective's derivative to 0.

$$\begin{aligned}
&\frac{\partial C(\theta_j)}{\partial \theta_j} = 0 \\
&= -2\sum_{t=1}^{T}(A_t - A)\left[(u_t - \bar{u}) - (A_t - A)^T\theta_j\right] \\
&= -2\sum_{t=1}^{T}\left[(A_t - A)(u_t - \bar{u}) - (A_t - A)(A_t - A)^T\theta_j\right]
\end{aligned}$$

Solving for θ_j,

$$\begin{aligned}
\theta_j^* &= \left[\sum_{t=1}^{T}(A_t - A)(A_t - A)^T\right]^{-1}\left[\sum_{t=1}^{T}(A_t - A)(u_t - \bar{u})\right] \\
&= \left[\left(\sum_{t=1}^{T}\frac{A_t A_t^T}{T}\right) - AA^T\right]^{-1}\left[\left(\sum_{t=1}^{T}\frac{A_t u_t}{T}\right) - \bar{u}A\right]
\end{aligned} \quad (17)$$

We call $\hat{u}_{\theta_j^*}$ the MIVAT estimator.

3.2 Multiplayer Games

For two-player, zero-sum games, we only need to learn a single estimator $\hat{u}_{\theta_1^*}$, since its negation is the variance minimizing estimator for the second player. In n-player, general-sum games ($n > 2$), we must learn individual θ_j for each position j. In the case of n-player, zero-sum games, however, the condition $\sum_{j=1}^{|N|}\hat{u}_{\theta_j^*}(z) = 0$ may not be satisfied when each of the functions are learned independently. We now show how to learn a set of parameters $\theta = \left[\theta_1^T, ..., \theta_{|N|}^T\right]^T$ satisfying the zero-sum constraint.

To satisfy the zero-sum constraint for a linear value function, it is sufficient[3] to require,

$$\theta_{|N|} = -\sum_{j=1}^{|N|-1}\theta_j.$$

[3]If the features are linearly independent on the sampled data, then it is a necessary condition as well.

To satisfy the zero-sum constraint, set $\theta_N = -\sum_{j=1}^{|N|-1}\theta_j$ and then only learn $\theta = (\theta_1, ..., \theta_{|N|-1})$. We can then optimize the column vector of unconstrained parameters $\theta = \left[\theta_1^T, ..., \theta_{|N|-1}^T\right]^T$ to minimize the sample variance averaged over all players.

Define $S_1, \ldots, S_{|N|}$ to be $n \times (|N|-1)n$ selection matrices such that:

1. $\forall\ j \in \{1, ..., |N|-1\}$, S_j has the identity matrix in the jth $n \times n$ column block of the matrix, and is zero elsewhere.
2. $S_{|N|}$ has the negative $n \times n$ identity matrix in all $|N|-1$ column blocks.

We can then write the target optimization as:

$$\underset{\theta \in \mathbf{R}^{(|N|-1)d}}{\textbf{Minimize:}} \; C(\theta) = \sum_{j=1}^{|N|}\sum_{t=1}^{T}\left[\begin{array}{c}(u_{t,j} - \bar{u_j}) \\ -(A_t - A)^T S_j\theta\end{array}\right]^2$$

where the selection matrices pick out the proper θ_j from θ.

As our objective is again convex in θ, we can solve this optimization by setting the objective's derivative to 0. Stepping through the derivation as before, we obtain the solution

$$\begin{aligned}
\theta^* &= \left[\sum_{j=1}^{|N|} S_j^T\left(\sum_{t=1}^{T}(A_t - A)(A_t - A)^T\right)S_j\right]^{-1} \\
&\qquad \left[\sum_{j=1}^{|N|}\sum_{t=1}^{T}(A_t - A)S_j(u_{t,j} - \bar{u}_j)\right] \\
&= \left[\sum_{j=1}^{|N|} S_j^T\left(\sum_{t=1}^{T}\frac{A_t A_t^T}{T} - AA^T\right)S_j\right]^{-1} \\
&\qquad \left[\sum_{j=1}^{|N|}\left(\sum_{t=1}^{T}\frac{u_{t,j}A_t}{T} - \bar{u_j}A\right)S_j\right]
\end{aligned} \quad (18)$$

The estimator for player j is then $\hat{u}_{S_j\theta^*}$, where $\sum_j \hat{u}_{S_j\theta^*}(z) = 0,\ \forall z \in Z$.

4 Applications to Poker

The MIVAT approach is applicable to any extensive game as well as frameworks subsumed by extensive games (e.g., finite-horizon MDPs and POMDPs). We explore its usefulness in the context of poker, a family of games involving skill and chance. We will first give background on Texas hold'em Poker, the basis for our three testing domains, and then describe how our approach was applied to these domains.

Texas Hold'em Poker. Texas hold'em is a card game for 2 to 10 players. A single hand consists of dealing two private cards to each player from a shuffled 52-card deck. The players act over the course of four rounds, in which each player in turn **calls** (matching the amount of money everyone else has put into the **pot**), **raises** (increasing the amount of money

everyone has to put into the pot), or **folds** (giving up the hand and losing whatever money they already placed into the pot). In between each round, public cards are revealed (3, 1, and 1 after rounds 1, 2 and 3, respectively). Among the players who have not folded after the fourth round, the player who can make the best five card poker hand from their cards and public cards wins the money placed in the pot. If all but one player folds, they win the pot regardless of their hand.

Texas hold'em can be played with fixed bet sizes (limit) or no fixed bet sizes (no-limit). We show results for two-player limit, two-player no-limit, and six-player limit poker. The remainder of this section explains how we applied MIVAT in each of these domains.

Implementation. We employed variations on five basic poker hand features. Hand-strength (HS) is the expectation over the yet-to-be-seen public cards of the probability of the evaluated hand beating a single other hand chosen uniformly at random. Hand-strength-squared (HS2) is the expectation over the yet-to-be-seen public cards of the squared probability of the evaluated hand beating a single other hand chosen uniformly at random. It is similar to hand-strength but gives a bonus for higher variance. Pot equity (PE), requiring knowledge of all players' hands, is the probability that the evaluated hand wins the pot assuming no additional player folds. The base features involved taking each of these hand-values for each player and multiplying them by pot size (PS), the amount of money in the pot, to make the features more representative of the potential utility (the final pot size).

In the two-player games, therefore, we had 7 base features. In order to allow for additional modelling power we took products of the basic features to get quadratic functions of the features instead of just linear functions. In some experiments in the two-player, limit domain, we included an additional feature: the hand-crafted value function used by DIVAT. We did this to illustrate that we can improve on well-designed value functions by using them as a feature. In the six-player game, we did not use the pot equity feature (due to its expensive computation[4]) or any feature cross-products.

5 Results

We examine the effectiveness of MIVAT using a number of different sources of poker hands. In all three domains we used data from programs playing against each other in the 2008 AAAI Computer Poker Competition (this involved nine agents for two-player limit, four for two-player no-limit, and six for six-player limit). We call this data "Bots-Bots". For the two-player limit domain, we also used a dataset involving a single strong poker program playing against a battery of weak to strong human players. We call this data "Bot-Humans". For each experiment we separated the data into 450,000 training hands and 50,000 testing hands, and then trained a linear value function using the MIVAT optimization. For each experiment we compare MIVAT's performance to that of Money (the estimator that just uses the player's winnings on each hand) and, when appropriate, DIVAT.

[4]In two-player games, these features were pre-computed in a table for all possible hand combinations. The two-player tables can be used for hand-strength and hand-strength-squared in a six-player game, but pot equity cannot.

Data	MIVAT	MIVAT+	DIVAT	Money
Bot-Humans	2.387	2.220	2.238	5.669
Bots-Bots	2.542	2.454	2.506	5.438

Table 1: Standard deviation of estimators for two-player, limit poker.

Data	MIVAT Bot-Humans	MIVAT Bots-Bots	DIVAT	Money
Bot-Humans	2.169	2.253	2.238	5.669
Limit-Bots	2.461	2.418	2.506	5.438

Table 2: Standard deviation of tailored estimators for two-player, limit poker.

Two-Player Limit Poker. We first compare the MIVAT optimization to the hand-crafted DIVAT estimator for two-player, limit poker. We computed two estimators: one with the DIVAT value as a feature (MIVAT+) and one without (MIVAT). Both estimators were trained using the Bot-Humans data. The standard deviation of these two estimators on testing data from both the Bot-Humans data and the Bot-Bot data is shown in Table 1. First, note that MIVAT without the DIVAT feature was still able to reach comparable variance reduction to the hand-crafted estimator, a dramatic improvement over the Money estimator. Further variance reduction was possible when DIVAT was used as a feature. MIVAT+'s improvement over DIVAT is statistically significant for the Bots-Bots data (even though it trained on the Bot-Humans data) but not for Bot-Humans.

MIVAT can also be used to train specific estimators for specific populations of players. We trained an estimator specifically for the Bot-Humans setting using the DIVAT feature and exploiting the fact that we knew whether the bot was the dealer on each hand. We also trained an estimator specifically for the Bots-Bots data also using the DIVAT estimator. Table 2 compares how well these functions and DIVAT perform on the two datasets: one they were trained for, and the other they were not. As expected, the functions perform better on their respective datasets than the more general functions from Table 1. Using the specific function, we now statistically significantly outperform DIVAT on the Bot-Humans dataset. Notice that the estimator trained specifically for the Bots-Bots dataset did not outperform DIVAT on the Bot-Humans data, suggesting that the estimators can be tailored to different populations of players.

Two Player No-Limit Poker. In two-player no-limit poker, the betting is only limited by a player's stack (their remaining money), which in the AAAI competition was kept at $1000. As a result the standard deviation of a player's winnings is generally much higher than in limit poker. MIVAT was trained using all cross-products of the standard features. Table 3 shows the resulting standard deviation on test data from the AAAI no-limit competition. MIVAT resulted in a

Data	MIVAT	Money
Bots-Bots	32.407158	42.339743

Table 3: Standard deviation of estimators for two-player, no-limit poker.

Data	MIVAT	Money
Bots-Bots	22.9234365	28.0091805

Table 4: Standard deviation of estimators for six-player, limit poker.

25% reduction on the standard deviation in a game where no hand-crafted value function was yet to be successful. The less dramatic reduction (as compared to limit) may be a result of players' strategies being inherently high variance, or more sophisticated features may be required. Features on the betting rather than just the cards or specific features distinguishing large-pot and small-pot games may result in additional improvements.

Six-Player Limit Poker. As in two-player, limit poker, there is no known reduced-variance estimator in six-player, limit poker. MIVAT was trained on 200,000 training hands and 20,000 test hands with six features: HS to the power of the number of non-folded players not folded. The standard deviation of MIVAT and the Money estimators are shown in Table 4. Again, MIVAT shows only a modest 20% variance reduction. The inability to match the success in two-player limit is likely due to the limited feature set (no pot-equity) and failure to capture the position of the players that had not yet folded, a key factor in a player's winnings. This result demonstrates that MIVAT does not completely remove the need for domain knowledge: informed identification of features could have considerable impact on the resulting estimator.

6 Conclusion

We introduced MIVAT, the Informed Value Assessment Tool, a general automated approach to obtaining low-variance, unbiased assessments of agent performance. We constructed an optimization for finding the variance-minimizing value function for use in an advantage sum estimator. We then derived a closed-form solution for the case of linear value functions. Our approach overcomes the requirement in previous work for hand-designed value functions, and can be used to tailor agent assessment to a known population distribution. We tested MIVAT on three variants of Texas hold'em poker: two-player limit, two-player no-limit and six-player limit poker. We showed that we can attain and even exceed the best existing hand-crafted esimator for two-player limit poker. We also showed that we can construct variance reducing estimators for two-player, no-limit and six-player, limit poker, where no previous estimators were available, based solely on simple poker features and logs of past experience.

There are two main avenues for future work. First, the MIVAT formulation focuses on optimizing a linear value function using variance as the loss. However, there are other options for the optimization. Expected variance is a convenient loss function, but like many squared-loss measures, it is heavily biased by outliers. An alternative loss may provide better estimators for typical agents, even while having a higher variance on average. Furthermore, optimizing non-linear value functions may alleviate the challenge of constructing good features. Second, we are also interested in extending the approach to complex settings where a fully explicit game formulation may not be available. For example, Robocup [Kitano *et al.*, 1997], TAC [Wellman *et al.*, 2003], and the General Game Playing Competition [Genesereth *et al.*, 2005] all have difficult evaluation problems, but it is not practical to fully specify their corresponding extensive game. One can still, however, identify the role of chance in these simulated settings as invocations of a random number generator. Luck terms with zero expectation could be computed (and a variance-minimizing value function optimized) using feature changes before and after the affects of the random number generator are applied. This approach could result in a guaranteed unbiased variance-reducing estimator and hence, more significant competition results.

7 Acknowledgments

We would like to thank the University of Alberta Poker Research Group for help throughout for ideas and implementation issues. This research was funded by Alberta Ingenuity, iCORE and NSERC.

References

[Billings and Kan, 2006] Darse Billings and Morgan Kan. A tool for the direct assessment of poker decisions. *International Computer Games Association Journal*, 29(3):119–142, 2006.

[Genesereth *et al.*, 2005] Michael R. Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the aaai competition. *AI Magazine*, 26(2):62–72, 2005.

[Kitano *et al.*, 1997] Hiroaki Kitano, Yasuo Kuniyoshi, Itsuki Noda, Minoru Asada, Hitoshi Matsubara, and Eiichi Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.

[Osborne and Rubinstein, 1994] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, July 1994.

[Wellman *et al.*, 2003] M. P. Wellman, A. Greenwald, P. Stone, and P. R. Wurman. The 2001 trading agent competition. *Electronic Markets*, 13:4–12, 2003.

[Wolfe, 2002] David Wolfe. Distinguishing gamblers from investors at the blackjack table. In *Computers and Games 2002, LNCS 2883*, pages 1–10. Springer-Verlag, 2002.

[Zinkevich *et al.*, 2006] Martin Zinkevich, Michael Bowling, Nolan Bard, Morgan Kan, and Darse Billings. Optimal unbiased estimators for evaluating agent performance. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, pages 573–578, 2006.