University of Alberta

A GENERAL FRAMEWORK FOR REDUCING VARIANCE IN AGENT EVALUATION

by

Martha White

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

©Martha White
Spring 2010
Edmonton, Alberta

Permission is hereby granted to the University of Alberta Libraries to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only. Where the thesis is converted to, or otherwise made available in digital form, the University of Alberta will advise potential users of the thesis of these terms.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Examining Committee

Michael Bowling, Computing Science

Dale Schuurmans, Computing Science

Peter Hooper, Mathematical and Statistical Sciences

Duane Szafron, Computing Science

To Mamina and Tati

For giving me so much love, support and opportunities throughout my life.

Abstract

In this work, we present a unified, general approach to variance reduction in agent evaluation using machine learning to minimize variance. 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 (sample averages); 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 based on control variates 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). This work has three main contributions. First, we consolidate previous work in using control variates for variance reduction into one unified, general framework and summarize the connections between this previous work. Second, our framework makes variance reduction practically possible in any sequential decision making task where designing the expert value function is time-consuming, difficult or essentially impossible. We prove the optimality of our approach and extend the theoretical understanding of advantage sum estimators. In addition, we significantly extend the applicability of advantage sum estimators and discuss practical methods for using our framework in real-world scenarios. Finally, we provide low-variance estimators for three poker domains previously without variance reduction and improve strategy selection in the expert-level University of Alberta poker bot. This work is an elaboration of published work [White and Bowling, 2009].

Acknowledgements

First and foremost, I would like to thank Michael Bowling, for being such a fantastic supervisor. Over these last four years, he has worked on several projects with me, patiently advising and encouraging me as well as answering numerous questions. He has provided an open and comfortable atmosphere for research and I cannot thank him enough. Second, I would like to thank the University of Alberta Poker Research Group, particularly Neil Burch and Morgan Kan. They have helped throughout with ideas and implementation issues. I would like to thank Dale Schuurmans for helping correct and complete ideas in the thesis. A special thanks to Amir massoud Farahmand for help with theoretical sections. Finally, I would like to thank my husband, Adam White, for always supporting me, for his clarity of thought and spending hours editing and discussing my thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Evaluating an agent's performance is a common task. It is a critical step in any agent development cycle: to improve the agent, a measure of its current skill is necessary. For example, a blackjack-playing agent may act optimally, receive an unlucky deal and lose value. The lost value suggests that the agent's strategy should be modified, but the action was actually optimal. Accurate evaluation is needed to properly improve the agent. Agent evaluation is also important for agent-agent interaction. For example, to adjust strategies against opponents, it is necessary to evaluate their skill against your strategy. Finally, objective agent (or algorithmic) comparison is necessary for ranking, such as in competitions, which have become popular in the artificial intelligence community. The Trading Agent Competition, Reinforcement Learning Competition and Poker Competitions all rely on the ability to accurately rank agents. To pursue worthwhile algorithmic avenues in research, it is necessary to have an accurate estimate of performance.

The most common approach for agent evaluation is *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. Since the estimate of the standard deviation is inversely proportional to the square root of the number of trials, the number of runs must be increased by a factor of 100 to reduce the error by a factor of 10.

For example, imagine pricing options for the stock market (i.e., estimating the value of stocks). Usually, a model is constructed to represent the expected changes in the option value. The model enables approximate predictions of the option's projected value and so provides a better estimate of its current value. Certain models have a closed form solution, such as the well-known Black-Scholes model. In many cases, however, the differential equations governing the model dynamics are too complex to solve in closed form due to the number of influencing variables. Instead, the value is estimated through Monte Carlo simulation: possible paths for the option are simulated and the resulting values averaged. For many option pricing problems, however, the number of samples required for accurate estimation can be prohibitively large [Lemieux and La, 2005]. In high-venture

environments, like option pricing, accurate estimation is pivotal. Since Monte Carlo estimation is commonly the only feasible option, variance reduction methods are key to precise evaluation.

The method of control variates is a powerful technique for variance reduction and has been studied in a number of fields, including statistics, operations research and game theory. Independently in operations research and game theory, an *advantage-sum* technique based on control variates was proposed for constructing a low-variance, unbiased estimate of 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 must 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 Chapter 3, we discuss these previous evaluation techniques and unify the work on control variates in these disjoint fields. With this insight, we construct a more general framework that facilitates the application of advantage sum estimators to a generic class of domains.

Our main contribution in this work is to develop a framework that uses machine learning to find a value function for the advantage-sum estimator. The estimator is guaranteed to be unbiased and, with a reasonable well-approximated value function, would reduce variance over the Monte Carlo estimate. Learning the value function 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 constant-sum and general-sum games. To learn the value function, 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 framework for learning the value function will be presented in Chapter 4.

In Chapter 6, we prove the optimality of our approach and illustrate that our optimization is an improvement over the optimization in previous advantage sum work. In Chapter 7, we illustrate

that our formalism encapsulates many domains outside finite extensive games and Markov chain processes (the previous domain classes for advantage sum estimators). In Chapter 5, we demonstrate the efficacy of the approach on the classic two-player limit poker game and three poker domains for which no previous variance reduction estimator exists: a two-player no-limit, a three-player limit and a six-player limit game. In addition, we provide a lower variance estimator with limited information in two-player limit, which better enables the University of Alberta poker bot to judge opponents while playing them and so adjust its strategy. Finally, in Chapter 8, we conclude this work, summarizing the contributions and discussing avenues for future work.

Chapter 2

Domain Formalism

In this chapter, we define the domain class for which we provide a general variance reduction framework. In Chapter 7, we will explore the broad scope of this formalism.

2.1 Sequential Decision Making Tasks

Definition 2.1.1 [1] *An **extensive game** Λ has the following components:*

- A finite set, $\{1, \ldots, N\}$, of **players**. Note that N can be 1.
- An **action set** $A \subseteq \mathbb{R}^k$ specifying all possible actions
- A bounded set of sequences, H, 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). We write $h \sqsubseteq z$ to denote that history h is a prefix of terminal history z. H and Z can be finite, countably infinite or uncountably infinite, but must be bounded to ensure that a terminal history is reached.
- $A(h) = \{a \in A : ha \in H\}$ are the actions[2] available after a non-terminal history $h \in H$. $A(h)$ must be measurable $\forall\, h \in H$.
- A **player function** P that assigns to each non-terminal history, $h \in H \backslash Z$, a member of $\{1, \ldots, N\} \cup \{c\}$, 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 distribution** $f_c(\cdot|h)$ on $A(h)$ ($f_c(a|h)$ is the probability that a occurs given h). If A is uncountable, then $f_c(\cdot|h)$ must be an integrable function $\forall h \in H$.
- 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 block 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.

[1] Adapted from the extensive game definition [p. 200][Osborne and Rubinstein, 1994]

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

- For each player $i \in \{1, \ldots, N\}$, a bounded utility function $u_i : Z \to \mathbb{R}$. If A is continuous, then u_i must be continuous $\forall\, i \in \{1, \ldots, N\}$.

A **strategy of player i**, $\sigma_i : A \times \{h \in H : P(h) = i\} \to [0, 1]$, in an extensive game is a function where $\sigma(a|h)$ is the probability that player i choses action a at history h. The strategy must respect the information sets: $\forall I_i \in \mathbf{I}_i,\ \forall h, h' \in I_i,\ \sigma_i(\cdot, h) = \sigma_i(\cdot, h')$. A **strategy profile** σ consists of a strategy for each player, $\sigma = (\sigma_1, \sigma_2, \ldots \sigma_N)$. The only assumption made in this work is that the game Λ (but not the players' strategies) is known. This assumption is common and we will discuss it further in Chapter 7, illustrating that it only minorly affects the scope of applicability.

2.1.1 Extensive Game Examples

Extensive games are a general model of sequential decision-making tasks. They encapsulate finite-horizon MDPS and POMDPs; finite extensive games, such as N-player general-sum or N-player constant-sum[3] games; and continuous actions decision tasks. MDPs are a widely used model, often used for tasks such as controllers and autonomous agents. Finite extensive games model multi-agent interaction, including domains such as poker, trading agents and multi-robot interaction. Finally, the continuous action settings that extensive games encapsulate are often used in robotics, such as for control of velocities and directions. Notable continuous action sequential decision making tasks include robotic arms for grasping and autonomous vehicles. We further discuss the generality of our domain class in Chapter 7.

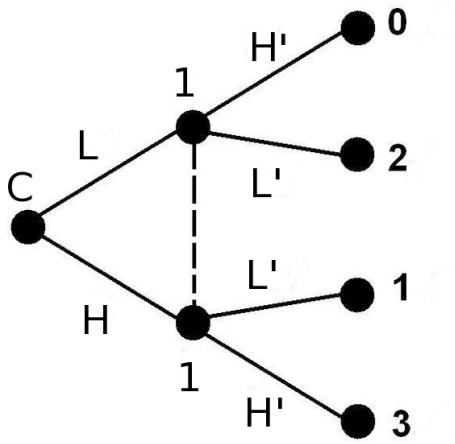

Figure 2.1: An extensive game with imperfect information.

To obtain a more intuitive understanding of extensive games, let us look at the example of an imperfect information, 1-player game with finite actions. In the game, the player guesses whether a deal is a high or low card. Figure 2.1 is the corresponding game tree. The card is dealt first: chance C is first to act. The dotted line represents that the player cannot see the cards, i.e., cannot see chance's action. Chance has two actions: deal a low card (L) or a high card (H). Player 1 has two actions: guess low (L') or high (H'). The terminal histories are $Z = \{LH', LL', HL', HH'\}$. We assume that chance deals low or high with equal probability: $f_c(L|\emptyset) = 0.5$ and $f_c(H|\emptyset) = 0.5$. There are four utility scenarios:

1. low deal with the player guessing high: $u(LH') = 0$

[3]Constant-sum games require that for some C, $\sum_{i=1}^{N} u_i(z) = C,\ \forall z \in Z$.

2. low deal with the player guessing low: $u(LL') = 1$

3. high deal with the player guessing low: $u(HL') = 2$

4. high deal with the player guessing high: $u(HH') = 3$

This is an imperfect information game because, before termination, a low and high deal appear equivalent to player 1: $\mathbf{I}_1 = \{\{L, H\}\}$.

In general, a game tree represents an extensive game by representing player ordering; all possible action sequences and their corresponding utilities; probability measures; and incomplete information. A game tree can similarly be constructed for continuous actions.

2.1.2 Agent Evaluation

The goal of **agent evaluation** is to estimate the expected utility of some player $j \in \{1, \ldots, N\}$ given a strategy profile, i.e.,

$$U_j = E_z \left[u_j(z) | \sigma \right]. \tag{2.1}$$

This expectation is dependent on the distribution $\Pr(z|\sigma)$, giving the probabilities of each outcome given the players are following strategy profile σ,

$$\Pr(z|\sigma) = \Pi_{\substack{ha \sqsubseteq z \\ P(h)=c}} f_c(a|h) \Pi_{\substack{ha \sqsubseteq z \\ P(h) \neq c}} \sigma_{P(h)}(a|h).$$

If the extensive game is small, one can compute this expectation exactly by enumerating the terminal histories for a given σ. For example, in the extensive game in Figure 2.1, given $\sigma(H'|L) = \sigma(H'|H) = 1$ (always chose H'), the expected value is

$$E_z \left[u_1(z) | \sigma \right] = 0.5 u_1(LH') + 0.5 u_1(HH') = 1.5$$

.

In large games, however, this approach is not practical. Chapter 3 illustrates different approaches for approximating this expectation to obtain a performance estimate for an agent. This thesis focuses on a general approach to improving the accuracy of this estimation problem.

Chapter 3

Performance Evaluation

Reducing variance in performance estimation has been a long researched topic in operations research, including methods such as common random numbers, antithetic variates, control variates, importance sampling, conditional Monte Carlo and stratified sampling (for an overview of these techniques, see [L'Ecuyer, 1994]). The technique that we use for our work is a special case of control variates, called advantage sum estimators, introduced to reduce variance specifically in extensive games [Zinkevich *et al.*, 2006] and separately for Markov chain processes [Dahl, 2001]. Independently, Dahl and Zinkevich discovered an approach to make construction of control variates simpler, when restricting the class to extensive games or Markov chain processes. This section begins with a discussion of the standard performance measure, Monte Carlo and then describes control variates and advantage sum estimators. Finally, we discuss several specific performance evaluators designed for four domains: blackjack, poker, the Trading Agent Competition and option pricing.

3.1 Monte Carlo Estimation

The common approach to agent evaluation is to estimate the expectation in Equation (2.1) 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,

$$\hat{U}_j = \frac{1}{T} \sum_t u_j(z_t). \tag{3.1}$$

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

$$\text{MSE}(\hat{U}_j) = \text{Var}\left[\hat{U}_j|\sigma\right] = \frac{1}{T}\text{Var}\left[u_j(z)|\sigma\right]. \tag{3.2}$$

This approach is effective when the domain has little stochasticity (i.e., $\text{Var}\left[u_j(z)\right]$ is small), trials are readily available (i.e., T can be made large), and/or the required precision is not small (i.e. large $\text{Var}[\hat{U}_j]$ 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 the

Monte Carlo 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,

$$\forall \sigma \quad E_z\left[\hat{u}_j(z)|\,\sigma\right] = E_z\left[u_j(z)|\sigma\right]. \tag{3.3}$$

In other words, Equation (3.3) 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 (3.1) to get an improved estimator. Many variance reduction tools focus on finding a low-variance $\hat{u}_j(z)$, enabling statistically significant conclusions to be drawn in a fewer number of trials. The next three sections summarize the background on using this approach for variance reduction. First, we explain the variance reduction techniques on which we base our framework: control variates and advantage sum estimators—an instantiation of the control variates method for extensive games. Then we summarize the previous work on constructing low-variance estimators for specific domains.

3.2 Control Variates

An intuitive technique for obtaining low-variance estimates is to try to extract more information from the given data: control variates are one approach to exploiting auxiliary information [L'Ecuyer, 1994]. The idea is to find control variates, random variables $C = (C_1, \ldots, C_n)^T$, that are correlated to the random variable X whose expectation, μ, we are approximating with Monte Carlo. Using this correlation and assuming that we know the expectation $E[C] = \nu = (\nu_1, \ldots, \nu_n)$, the variance of the estimator can be reduced using the controlled estimator:

$$X_c = X - \beta^T(C - \nu) = X - \sum_{k=1}^{n} \beta_k (C_k - \nu_k)$$

with $\beta = (\beta_1, \ldots, \beta_n)^T$ a vector of weights on the control variates. In the following, we illustrate the unbiasedness of the controlled estimator and the relationship between the controlled and Monte Carlo estimate.

If we let $\Sigma_C = \mathrm{Cov}[C]$ and $\sigma_{XC} = (\mathrm{Cov}[X, C_1], \ldots, \mathrm{Cov}[X, C_n])^T$, then we can see that:

$$E[X_c] = E[X] - \beta^T E[C - \nu] = E[X] \quad \text{(unbiasedness)}$$
$$\mathrm{Var}[X_c] = \mathrm{Var}[X] + \beta^T \Sigma_C \beta - 2\beta^T \sigma_{XC}$$

The variance is minimized with $\beta = \beta^* = \Sigma_C^{-1}\sigma_{XC}$, giving $\mathrm{Var}[X_c] = (1 - R^2_{XC})\mathrm{Var}[X]$, where $R^2_{XC} = \frac{\sigma_{XC}^T \Sigma_C^{-1} \sigma_{XC}}{\mathrm{Var}[X]}$ is the coefficient of determination. The variance, therefore, can be

reduced by either positive or negative correlation, the magnitude of the variance reduction indicated by the coefficient of determination. Notice that if R_{XC} is ± 1, the variance is reduced to zero; if there is no correlation, the variance is the same as the Monte Carlo estimator's variance.

To obtain this variance reduction in practice, the weights β are usually approximated with the sample covariance matrices: $\hat{\beta} = \hat{\Sigma}_C^{-1}\hat{\sigma}_{XC}$. If the covariance matrices are estimated on separate data (i.e. independent samples), the estimator is unbiased and should reduce variance over the Monte Carlo estimate. Typically in the control variates literature, however, the covariance matrices are estimated on the same samples used for the performance estimate, making the estimator biased. To see why, let $X_{c,t} = X_t - \hat{\beta}(C_t - \nu)$ where X_t, C_t are the tth samples of X, C with $t = 1, \ldots, T$. Let $\overline{X}_c$ and s_c^2 be the sample average and sample variance of those $X_{c,t}$. Recall that $E[XY] \neq E[X]E[Y]$ if random variables X and Y are dependent. Then we can see that

$$\begin{aligned}
E[\overline{X}_c] &= E\left[\sum_{t=1}^{T} X_t - \hat{\Sigma}_C^{-1}\hat{\sigma}_{XC}(C_t - \nu)\right] \\
&= E[X] - E\left[\hat{\Sigma}_C^{-1}\hat{\sigma}_{XC}\sum_{t=1}^{T}(C_t - \nu)\right] \\
&\neq E[X] - E\left[\hat{\Sigma}_C^{-1}\hat{\sigma}_{XC}\right]\underbrace{E\left[\sum_{t=1}^{T}(C_t - \nu)\right]}_{=0} \\
&= E[X]
\end{aligned}$$

because $\hat{\Sigma}_C^{-1}\hat{\sigma}_{XC}$ and $\sum_{t=1}^{T}(C_t - \nu)$ are estimated with the same samples and so are dependent. If $\hat{\Sigma}_C$ and $\hat{\sigma}_{XC}$ were estimated with independent samples $\{T+1, ..., n\}$, the '$\neq$' turns into an '$=$' above and the estimate would be unbiased.

Fortunately, if the controlled estimator is biased, the bias will likely increase the variance only for small T. Asymptotically, as $T \to \infty$, the mean squared error of $\overline{X}_c$ is smaller than $\overline{X}$, i.e. $\mathrm{Var}(\overline{X}_c) + (\mathrm{Bias}(\overline{X}_c, \mu))^2 < \mathrm{Var}(\overline{X}) + (\mathrm{Bias}(\overline{X}, \mu))^2 = \mathrm{Var}(\overline{X})$ [Nelson and Richards, 1990]. In cases with small T, techniques like jackknifing and splitting can be used to reduce the bias (see [Avramidis and Wilson, 1993] and [Nelson and Richards, 1990]).

Though control variates have enabled successful variance reduction in several applications, their general application has been hindered due to the need to find control variates *with known expectation*. In most cases, the expectation of the control variates has been estimated, which adds bias (see Section 3.5.3). In other cases, restricting the domain class has enabled a generic control variate. This approach has been taken for extensive games with the advantage sum framework and for Markov chain processes. With our work, we further facilitate construction of control variates for a general domain class.

3.3 Advantage Sum Estimators

Recently, Zinkevich and colleagues [Zinkevich *et al.*, 2006] developed a specification of control variates for finite extensive games. By focusing on finite extensive games, they make construction of the control variates simpler and so make the technique more practically applicable. Their low-variance estimators are called **advantage sum estimators**: this section describes the advantage sum formalism on which we build our system and which we expand to extensive games.

Assume one is given a real-valued function on histories $V_j : H \rightarrow \Re$ such that $V_j(z) = u_j(z),\ \forall z \in Z$, for a given position $j \in \{1, \ldots, N\}$. Define the following real-valued functions on terminal histories,

$$S_{V_j}(z) = \sum_{\substack{ha \sqsubseteq z \\ P(h) \neq c}} V_j(ha) - V_j(h) \tag{3.4}$$

$$L_{V_j}(z) = \sum_{\substack{ha \sqsubseteq z \\ P(h) = c}} V_j(ha) - V_j(h) \tag{3.5}$$

$$P_{V_J} = V_j(\emptyset). \tag{3.6}$$

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. In this way, S_{V_j} is a joint skill: it does not represent a particular players global or overall skill but rather its skill *when playing with these specific players*.

The advantage sum estimator is now simply,

$$\hat{u}_{V_j}(z) = S_{V_j}(z) + P_{V_j}(z). \tag{3.7}$$

We now need to investigate the conditions under which the skill plus position is an unbiased estimator. Because terms cancel when summing skill, luck, and position, we can rewrite the utility as a function of the skill, luck and position.

$$\begin{aligned} S_{V_j}(z) + L_{V_j}(z) + P_{V_j} = & \\ &= V_j(\emptyset) \\ &\quad + V_j(a_1) - V_j(\emptyset) \\ &\quad\quad + V_j(a_1 a_2) - V_j(a_1) \\ &\quad\quad\quad \ldots + V_j(z) - V_j(a_1 \ldots a_{|z|-1}) \\ &= V_j(z) \\ &= u_j(z) \end{aligned}$$

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:

$$\begin{aligned} E\left[\hat{u}_{V_j}(z)|\sigma\right] &= E\left[S_{V_j}(z) + P_{V_j}(z)|\sigma\right] \\ &= E\left[S_{V_j}(z) + P_{V_j}(z) + L_{V_j}(z)|\sigma\right] \qquad \ldots \text{if } E\left[L_{V_j}(z)|\sigma\right] = 0 \\ &= E\left[u_j(z)|\sigma\right] \end{aligned}$$

In particular, they suggest using the expected utility from a given history, h, to guarantee the zero luck constraint: $V_j(h) = E\left[u_j|h,\sigma\right]$. Chapter 6 reiterates their optimality results justifying this choice. We suggest a different choice for V_j, illustrating that it is an more appropriate when learning approximations for V_j.

3.4 Markov Chain Processes

Independently, a very similar advantage sum framework was discovered for Markov chain processes [Dahl, 2001]. A Markov chain process $\{X_k\} \subset H$, is a random process with given initial state $X_0 \in H$, governed by the transition probabilities $P(X_{k+1} = y|X_k = x)$. The term *Markov* means that the current state fully captures any information influencing future states; the probability function, therefore, relies solely on the current state and is independent of past states. This section briefly explains any differences arising from using Markov chain processes as the domain class, making it clear how we unify previous work into one general framework.

To enable evaluation for Markov chain processes, two assumptions are needed. First, we must assume that there exists a set of terminal histories $Z \subset H$ in which the process X stops and which X reaches with probability 1. Secondly, we must assume that we have a function $u : Z \to \mathbb{R}$ for which we are estimating the expectation. With these additions, we can build a similar framework to the advantage sum framework. The expected change to $V(h)$ is computed at each point in the history because there are no agents; there is only a position term for the initial X_0 and no skill term. Interestingly, like with advantage sum estimators, they suggest that the value function predict the utility, i.e. $V(h) = E\left[u|h,\sigma\right]$. Using this suggestion, they test their framework on a toy combat application, obtaining a variance reduction factor of approximately 1660 using a neural network to learn the value function predicting the utility.

There remain two subtle differences that can easily be overcome to unite the two frameworks. First, Markov chains require the Markov property. We can, however, circumvent this difference by noting that an extensive game becomes Markov if each state includes the complete history from all the previous states. Second, Markov chains do not explicitly mention agents and assume that transition probabilities are known on each time step. Though we cannot compute $Pr(X_{k+1} = y|X_k = x)$, X_{k+1} a player action, without knowing the strategies, we can ignore the player actions

using $\hat{u}_V(z) = S_V(z) + P_V(z) = u(z) - L_V(z)$. Because of the Markov property, we will still be able to compute $Pr(X_{k+2} = y | X_{k+1} = x)$, with X_{k+1} a player action and X_{k+2} a chance action, which is all that is need to compute $L_V(z)$. In addition, their framework can easily incorporate the multi-player setting by using a vector utility function $\mathbf{u} = (u_1, \ldots, u_N)$.

Despite some minor differences, the frameworks designed for extensive games and Markov chain processes are essentially identical; for the remainder of this thesis, when mentioning advantage sum estimators, we are referring to both the work in extensive games and in Markov chain processes.

3.5 Evaluators for Specific Domains

There have been several low-variance analysis tools developed for specific domains: blackjack, poker, the Trading Agent Competition and option pricing. In particular, each is specific to its own domain; none provide a general analysis tool. This section briefly explains the different approaches taken by these previous tools; the following chapters will illustrate how we go beyond this previous work to provide a more general technique for creating low-variance agent analysis tools.

3.5.1 Blackjack

Wolfe [Wolfe, 2002] reduced variance in performance evaluation for blackjack by indirectly using the idea of control variates. Blackjack is a imperfect information, one player card game with high variance. We can obtain significant variance reduction in blackjack by computing a near-optimal baseline strategy for comparison (possible because blackjack is a relatively small game). The expected value of any possible sequence of player actions can be very accurately approximated because of the single player scenario and relatively small number of moves in one game. Correspondingly, a near-optimal strategy can be computed as the baseline policy. The player's winnings on a given hand are compared with the winnings the baseline policy would have attained had it been employed with the same sequence of deals. The resulting unbiased estimator is this difference added to the baseline policy's known expected winnings (computed analytically). 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 because the expectation of an unbiased baseline policy is difficult to compute without knowing the other players' strategies.

3.5.2 Texas Hold'em Two-Player Limit Poker

There has been a significant amount of work on reducing variance for agent evaluation in Texas hold'em two-player limit poker. The first tools introduced included the Expected Value Assessment Tool (EVAT) and the Luck Filtering Analysis Tool (LFAT), an improvement on EVAT. Both of these tools, however, were based on heuristic approaches for reducing variance, rather than on a mathematically justified framework such as control variates. The first tool that enabled low-variance,

unbiased estimators in Texas hold'em two-player limit poker was the Ignorant Value Analysis Tool (DIVAT) [Billings and Kan, 2006].

DIVAT uses the advantage sum framework with a hand designed function shown to satisfy the zero-luck constraint. Similar to blackjack, they construct a baseline policy for comparison, but use it slightly differently. They estimate the skill of a player on any round in the game using the difference between the expected value of the player's action and the baseline's value on that round, assuming both play the baseline policy from that point to complete the game. These differences are summed for all rounds to obtain a gain or loss over the baseline policy. This estimate is an advantage sum estimate where the value function is the expected value of the current history assuming that both players follow the baseline strategy to a terminal history. From careful design of the baseline policy, the estimate is provably unbiased. Because we are now in a two-player game, the baseline policy is more difficult to define as we cannot compute a near optimal strategy. They construct a baseline policy by essentially splitting up a player's strategy based on aggression thresholds, optimized empirically. For a more in-depth discussion, read Billings and Kan [2006] or Kan [2007].

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. Augmenting DIVAT with importance sampling [Bowling *et al.*, 2008] and common random numbers[1], the University of Alberta Poker group were able to draw statistically significant conclusions about their performance against expert human players in only 500-1000 hands. For example, in both the first and second Man-vs-Machine matches, none of the results were statistically significant using Monte Carlo. With DIVAT, they could statistically significantly conclude that in the first Man-vs-Machine match against two-expert human players, they had two wins and two draws. In the second Man-vs-Machine against six expert human players, they were able to conclude that they had three wins and three draws.

Though DIVAT has been successful in two-player limit poker, the hand-designed value function does not extend to other games (even other poker games). Specifically, no useful reduced-variance estimator has been identified for no-limit poker or games involving more than two players.

3.5.3 Trading Agent Competition

The Trading Agent Competition for Supply Chain Management (TAC/SCM) is a market simulation in which autonomous agents act as manufacturers in a two-tier supply chain marketplace. Agents purchase components from the supplier, manufacture a particular finished product and compete with each other to sell these products to customers in a reverse auction. Stochasticity in the simulation occurs from probability distributions on availability of supplies and customer demand. The performance of an agent is an average of their profit for each round of the reverse auction. This estimate,

[1]The University of Alberta Poker group ran *duplicate* matches for the Man-vs-Machine poker matches by running two instances of the program against two human players. If the first human player and bot received C1C2 and C3C4 respectively in the deal, then the second human player and bot would de dealt C3C4 and C1C2 respectively

however, has produced statistically insignificant ordering of the agents since the first competition in 2002.

To reduce the variance of this estimate, Estelle et al. define a control variate based on the known underlying stochastic process: the demand [Estelle *et al.*, 2004]. With only a specification for the stochastic demand process and no closed-form probability density function, however, they cannot compute the mean of the control variate. Instead, they estimate the density function of the demand using Monte Carlo samples of demand trajectories over simulated games with a kernel-based density estimation method. The resulting estimator is slightly biased due to approximations in the density function; they illustrate, however, that in realistic scenarios, the contribution of bias to the MSE is small compared to the variance. They obtain a variance reduction factor of up to 1.4, decreasing the number of required samples by up to a factor of 2. The difficulty in constructing control variates and the relatively small amount of variance reduction in TAC illustrates the likely benefit of a general purpose variance reduction framework facilitating construction of control variates.

Note that a similar approach to variance reduction was also used in the Trading Agent Competition Classic, where agents act as travel agents designing travel packages for customers. In addition, like in poker, variance was further reduced with common random numbers [Sodomka *et al.*, 2007].

3.5.4 Option pricing

Another field where control variates has had an important influence is option pricing. An option is a contract between a buyer and seller stating that a buyer has the option to either buy or sell a particular asset for an agreed upon amount. Pricing options is a difficult task in a complex, dynamic environment, in particular for real time trading where time is restrictive. Variance reduction techniques can be crucial for producing an acceptably accurate price in a limited amount of time. Three methods for using control variates have been proposed for option pricing: ordinary control variates, a replicating delta hedge and re-simulation [Lidebrandt, 2007]. Here, we will only discuss the use of ordinary control variates.

Lidebrandt suggests two options to be used as control variates: an Asian option or a Cliquet option. In order to use these options as control variates, their expected values need to be calculated. In the case of a few types of Asian options, the expected value of the option can be exactly computed and for others reasonably approximated. For the more complicated Cliquet option, the approximation is computationally restrictive and more complex to compute. In combination with their other methods, in particular re-simulation, they obtain a significant speed-up (in one case, 500 times shorter simulation time using some Cliquet options); the ordinary control variates on their own, however, except in very simplified scenarios, decrease the variance minorly.

Interestingly, the idea of using comparative options is similar to the idea of using a baseline policy. Intuitively, a reasonable predictor to compare against seems like it should remove a significant amount of stochasticity: the reasonable predictor should be similarly affected by stochasticity. The

work in options pricing, however, again illustrates the difficulty in designing this baseline policy. In the remainder of this thesis, we present a general approach to constructing control variates and illustrate its theoretical and empirical advantage.

Chapter 4

MIVAT

A more general approach to using the advantage sum estimator from Equation (3.7) is to automatically construct 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 chapter describes the MIVAT approach, a framework for learning the value function in extensive games. First, we define a simplified approach to learning the value function and present a novel optimization to find the ideal value function. We then derive a closed form solution for this novel optimization and provide an example of how constraints can be enforced on the optimization by finding a closed form solution for N-player constant-sum games. Note that for the presentation, we assume that we have a discrete set H and so, to estimate expectations, use sums rather than integrals. This choice is for clarity of the presentation. The final section will discuss how these results remain applicable for continuous actions.

4.1 Learning the Value Function

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

$$\hat{u}_{V_j}(z) = u_j(z) - L_{V_j}(z). \tag{4.1}$$

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.[1] We then define V_j for the remaining histories, where chance is next to act, to explicitly satisfy the

[1] 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.

zero-luck constraint [2],

$$V_j(h \text{ s.t. } P(h) = c) \equiv \sum_{a' \in A(h)} f_c(a'|h) V_j(ha'), \tag{4.2}$$

so then,

$$L_{V_j}(z) = \sum_{\substack{ha \sqsubseteq z \\ P(h)=c}} \left(V_j(ha) - \sum_{a' \in A(h)} f_c(a'|h) V_j(ha') \right), \tag{4.3}$$

giving

$$E[L_{V_j}|\sigma] = \sum_{\substack{ha \sqsubseteq z \\ P(h)=c}} \left(E[V_j(ha)|h] - E[\underbrace{\sum_{a' \in A(h)} f_c(a'|h) V_j(ha')}_{=E[V_j(ha)|h]}] \right) = 0. \tag{4.4}$$

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 almost unconstrained in our choice of value function. In the discrete case, we are unconstrained in our choice of value function. In the continuous state case, the value function must be an integrable function (which is not a strong restriction).

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.

$$\underset{V_j}{\textbf{Minimize:}} \quad \sum_{t=1}^{T} \left(\hat{u}_{V_j}(z_t) - \frac{1}{T} \sum_{t'=1}^{T} \hat{u}_{V_j}(z_{t'}) \right)^2 \tag{4.5}$$

4.2 Derivation of Linear Optimization

In order to make the optimization in Equation 4.5 tractable, we focus on the case where V_j is a linear function. Let $\phi : H \to \mathcal{R}^d$ map histories to a vector of d features. We require,

$$V_j(h) = \phi(h)^T \theta_j, \tag{4.6}$$

for some $\theta_j \in \mathcal{R}^d$. We will use $\hat{u}_{\theta_j}$ as shorthand for the advantage-sum estimator, $\hat{u}_{V_j}$ from Equation (4.1), with V_j defined as the linear function above.

[2] This explicit formulation to satisfy the zero-luck constraint is used in both the extensive games and Markov chain work, though presented slightly differently

We now derive a closed-form solution, θ_j^*, to our optimization. Let $u_t = u_j(z_t)$. From Equations (4.1) and (4.3), we get,

$$\hat{u}_{\theta_j}(z_t) = u_t - \left[\sum_{\substack{ha \sqsubseteq z_t \\ P(h)=c}} \left(\phi(ha) - \sum_{a' \in A(h)} f_c(a'|h)\phi(ha')\right)\right]^T \theta_j \tag{4.7}$$

Define the following shorthand notation,

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

$$A = \frac{1}{T}\sum_{t=1}^{T} A_t \qquad \overline{u} = \frac{1}{T}\sum_{t=1}^{T} u_t$$

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

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

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

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

$$\begin{aligned}
\theta_j^* &= \left[\sum_{t=1}^{T}(A_t - A)(A_t - A)^T\right]^{-1}\left[\sum_{t=1}^{T}(A_t - A)(u_t - \overline{u})\right] \\
&= \left[\left(\sum_{t=1}^{T}\frac{A_t A_t^T}{T}\right) - AA^T\right]^{-1}\left[\left(\sum_{t=1}^{T}\frac{A_t u_t}{T}\right) - \overline{u}A\right]
\end{aligned} \tag{4.8}$$

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

4.3 Extension for Constant-Sum Games

We have derived a closed-form solution for the MIVAT estimator in the linear case, but have not allowed any constraints on the optimization which in practice may be important for the success of the estimator. In this section, we illustrate how to impose the constant-sum constraint for extensive games on the MIVAT estimator. This derivation demonstrates at least one scenario in which the

MIVAT estimator has a closed form solution with constraints and hopefully suggests a method to add other desired constraints to the optimization.

For two-player, constant-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, constant-sum games, however, the condition $\sum_{j=1}^{N} \hat{u}_{\theta_j^*}(z) = c$ may not be satisfied when each of the functions are learned independently.

To satisfy the constant-sum constraint for an estimator with a linear value function $\theta = \left[\theta_1^T, \ldots, \theta_N^T\right]^T$, we need

$$\begin{aligned} c &= \sum_{j=1}^{N} \hat{u}_{\theta_j}(z_t) \\ &= \sum_{j=1}^{N} u_j(z_t) - A_t^T \theta_j \\ &= \left(\sum_{j=1}^{N} u_j(z)\right) - \left(\sum_{j=1}^{N} A_t^T \theta_j\right) \\ &= c - \left(\sum_{j=1}^{N} A_t^T \theta_j\right) \end{aligned}$$

It is therefore sufficient, and, if the features are linearly independent on the sampled data, necessary to require,

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

To satisfy this constraint, set $\theta_N = -\sum_{j=1}^{N-1} \theta_j$ and then only learn $\theta = (\theta_1, \ldots, \theta_{N-1})$. We can then optimize the column vector of unconstrained parameters $\theta = \left[\theta_1^T, \ldots, \theta_{N-1}^T\right]^T$ 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. $\forall\, j \in \{1, \ldots, N-1\},\ S_j$ 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:

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

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

$$\theta^* = \left[\sum_{j=1}^{N} S_j^T \left(\sum_{t=1}^{T}(A_t - A)(A_t - A)^T\right) S_j\right]^{-1} \left[\sum_{j=1}^{N}\sum_{t=1}^{T} S_j^T (A_t - A)(u_{t,j} - \overline{u}_j)\right]$$
$$= \left[\sum_{j=1}^{N} S_j^T \left(\sum_{t=1}^{T} \frac{A_t A_t^T}{T} - AA^T\right) S_j\right]^{-1} \left[\sum_{j=1}^{N} S_j^T \left(\sum_{t=1}^{T} \frac{u_{t,j} A_t}{T} - \overline{u}_j A\right)\right] \tag{4.9}$$

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

4.4 Continuous Action Space

The above results were presented assuming we had a discrete set of histories, H. The results still hold, however, if we consider continuous state spaces. The important factor is knowledge of the chance node dynamics, enabling us to define V_j to satisfy the zero luck constraint,

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

Now, instead of the discrete expectation $\sum_{a' \in A(h)} f_c(a'|h) V_j(ha')$, we need the (Lebesgue) integral $\int_{A(h)} f_c(a|h) V_j(ha) da$. Notice that since the product of two integrable functions is integrable, we need only require that $f_c(\cdot|h), \forall\, h \in H$ and V_j are integrable to ensure that the MIVAT estimator is well-defined. As was shown in the discrete case, if $|A(h)|$ is reasonably small in the discrete case $\forall\, h \in H$, then we can exactly compute the expectation. Otherwise, if $|A(h)|$ is in-feasibly large or infinite, and the expectation unknown, then the expectation has to be approximated. Similarly in the continuous case, if we have a closed form solution for the integral, then we can exactly compute $\int_{A(h)} f_c(a|h) V_j(ha) da$ for each history $h \sqsubseteq z_t$. Without a closed form solution for the integral, however, it must be approximated. We will briefly discuss situations where the integral has a closed form solution and, if there is no closed form solution, approaches for approximating the integral.

For certain forms of the value function and probability measures $f_c(\cdot|h)$, we can guarantee a closed form solution for the integral. Due to the large scope of the topic, we will only briefly discuss possible options to be illustrative rather than complete. First, let us assume we are using a linear value function. Because of the linearity of integration, we need only require that we have features ϕ and probability measures f_c such that $f_c(\cdot|h) \circ \phi(h\cdot)$ is integrable $\forall h \in \{h \in H | P(h) = c\}$. To obtain a closed form solution, however, we need to make some assumptions about ϕ and f_c. Most simply, if we assume that the features and probability measure follow a Gaussian distribution, then

the product of two Gaussians is again a Gaussian, giving a solution for the integral. If we only want to assume that the dynamics are Gaussian, there are several limited forms for the features, including forms using a^n, sin^n, cos^n and ln (see [Wikipedia, 2009a] for a more complete list of exponential function integrals). The feature form a^n, i.e., polynomial on the feature, is a common approach to increasing modeling power of the linear value function. If we want to look at other distributions for the dynamics, we can similarly look at known antiderivatives for integrals involving that distribution function. In some scenarios, however, the limitations on the feature functions due to the choice of distribution may too negatively impact the variance reduction of the estimator. In this case, it may be desirable to find a numerical approximation of the integral, rather than requiring a closed form solution.

For the more general case with an unknown integral value, there are a host of integral approximation techniques for definite integrals. Recall that a definite integral is an integral over an interval $[a\ b]$; in practice, its common for $A(h)$ to be an interval because each real-value corresponds to choosing a state and because A must be bounded. If the function $f_c(\cdot)V_j(h\cdot)$ can be written explicitly, there are several techniques for estimating the integral, including divide and conquer, dyadic decomposition (a special case of divide and conquer) and dimensional analysis [Tricki, 2009]. If the function cannot be represented explicitly, then the integral can be approximated based on sample points. There is a huge literature on numerical approximations for a definite integral, including (1) the Newton-Cotes formulas such as the midpoint rule, the trapezoidal rule and Simpson's rule; (2) the typically more accurate quadrature rules such as Gaussian quadrature, Clenshaw-Curtis quadrature and Gauss-Kronrod quadrature; and (3) Monte Carlo integration [Wikipedia, 2009b]. There are many software packages that provide efficient implementations of these methods.

The integral approximation techniques rely on the number of samples n provided; there are several methods for using samples more effectively for more accurate estimates. For example, Monte Carlo integration uses uniform sampling; two adaptive algorithms that improve results over Monte Carlo integration use importance sampling and stratified sampling. Richardson extrapolation is used to accelerate the convergence of a sequence; it modifies an $O(h^n)$ estimation to a $O(h^m)$ estimate where $m > n$. In general, many variance reduction techniques could be used to improve the sample estimate of the expectation $\int_{A(h)} f_c(a|h)V_j(ha)da$.

Chapter 5

Results

In the previous chapter, we developed the MIVAT framework. In this chapter, we explore its usefulness in the context of poker, a family of games involving skill and chance. We will first briefly explain the rules of the game and give an illustrative example of why poker is representative of domains requiring variance reduction. Then, we will describe the features used for poker and present results in four poker domains: two-player limit, two-player no-limit, three-player limit and six-player limit poker. We illustrate that, with basic features, we are able to outperform DIVAT, which has been highly successful in two-player limit poker. Moreover, we illustrate that by tuning a value function to particular players, we can further reduce variance. We then illustrate significant variance reduction in the other three poker domains, for which its has been difficult to design a value function. Finally, we also reduce variance in two-player limit poker when only one of the player's cards is known (two-player limit poker with limited information). This result allows for lower variance estimation of performance during play, which can be used in a process that adapts the agent's strategy to its opponent's strategy.

5.1 Texas Hold'em Poker

Texas hold'em poker is an illustrative game of skill and luck where high variance in outcome can greatly hinder proper assessment. This section explains the rules of Texas hold'em poker and why we have chosen it as a motivating domain.

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 maximum amount of money anyone 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 betting 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 betting sizes (limit) or no fixed betting sizes (no-limit). Each In limit Texas hold'em poker, there is a limit of four raises per round of a fixed amount. The fixed raise amount in the first two rounds is called the **small bet**, which is doubled in the last two rounds and called the **big bet**. In no-limit, each round can have any number of raises of any amount greater than or equal to the minimum bet, bounded by each player's amount of money (set to some maximum stack size). Consequently, no-limit poker has much higher variance in the Monte Carlo estimate than limit poker. In this thesis, we illustrate the application of our analysis tool in three poker domains: two-player limit, two-player no-limit, and six-player limit poker.

The following example illustrates the luck and skill in two-player limit Texas hold'em poker and motivates the need for a value analysis tool in poker. The motivating example is from Morgan Kan's thesis on DIVAT [Kan, 2007]. Suppose Alice is dealt a **T**♠−**6**♠ and Bob a **J**♢−**5**♢ with Bob first to act. Though his hand is not particularly strong, it has potential and so he calls. Alice, with a similar strength hand, checks. The flop, **T**♢−**5**♡−**T**♡ gives Alice an extremely strong hand of three of a kind (3 tens, **T**). Bob has a pair (two fives) and likely to have the best hand. Deceptively, Alice checks to show weakness; Bob bets with his reasonable hand. Since Bob has shown some strength, Alice now raises without fear of Bob folding and thus losing her strong hand. Bob simply calls, possibly because Alice's move was unexpected and he is caught off guard. Bob receives a *flush draw* on the turn with a **2**♢, giving him the chance to obtain a stronger hand than Alice if he receives a ♢ on the river. Alice bets her strong hand and Bob raises, because he believes he still might have the best hand (especially with a flush-draw) and potentially because Alice is an aggressive player who often check-raises with marginal hands. Alice re-raises and Bob calls due to an already large pot and his flush draw, though he suspects he is beat. The river is the **2**♡, making it possible for Alice to lose to only one hand (**2**♠−**2**♣). Alice bets and Bob calls because the pot is already large and folding would be a big mistake. Alice wins the 22 small bets in the pot.

Now the question is: who was the better player in this game? Alice seems to be the stronger player, but maybe was mostly lucky to receive a stronger hand than Bob's also seemingly strong one, which caused the pot to get large. Did Alice make as much money as possible under the circumstances? Did Bob play as well as he could? These questions are difficult to answer by only looking at the winnings, which suggest that Alice was an exceptional player.

We can see that poker is a prime example of a domain requiring variance reduction for accurate performance estimates. As we have seen in this thesis, with a good value function, we can decrease the variance significantly. Learning a linear value function, we obtain considerable variance reduction in three poker domains. The next section explains the feature choices for the linear value function and then we present results for the three poker domains.

5.2 Feature Selection

We employed variations on five basic poker hand features. Hand-strength (HS) is the expected probability over the yet-to-be-seen public cards of the evaluated hand beating a single other hand chosen uniformly at random from the remaining cards. Hand-strength-squared (HS2) is the second moment of HS: the expected 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 these 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 it as a feature. In the six-player game, we did not use the pot equity feature due to its expensive computation[1].

Finally, we also learned separate functions for different betting scenarios. The idea of splitting on the betting is that players in specific betting scenarios should react similarly to previously being in this scenario, making the data on this split lower variance. The possible betting options are **check** (k) if there are no raises to **call** (c), **bet** (b) to initiate a bet when no raises are pending and **raise** (r) to increase a previous bet. For example, the first player may place a bet of two small bets, the second player will raise by two small bets and the first player will call, giving the sequence **brc**. If the first player does not wish to bet, he checks, the second player may then bet and the first player call, giving the sequence **kbc**. Betting scenarios for two-player limit poker included the following splits:

1. $A = \{kk\},\ B = \{bc, kbc, brc, kbrc, brrc, kbrrc, brrrc, kbrrrc\}$ (i.e. everything else)
2. $A = \{kk\},\ B = \{bc, brrc, kbrc, kbrrrc\},\ C = \{brc, brrrc, kbc, kbrrc\}$
3. $A = \{kk\},\ B = \{bc, kbc\},\ C = \{brc, kbrc\},\ D = \{brrc, brrrc, kbrrc, kbrrrc\}$

The first split separates out complete passive play from some agression. The second splits the betting based on which player last raised. The third splits the betting based on how much goes in the pot, with the most and second most being grouped together. We only used the betting-splits in two-player limit poker with restricted information, not for the general estimator compared against DIVAT. We found that the betting splits had no impact over the significant variance reduction already being obtained in two-player limit poker.

[1] 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.

For two-player no-limit poker where there is not a finite number of possible betting sequences, we split the data based on the pot-size. For example, we learned a different function for data with a potsize below 5; between 5 and 15; between 15 and 40; between 40 and 70; between 70 and 100; between 100 and 130; and finally above 130. We tried many different splits and found this split to be the best for our no-limit data. The stack size was 1000.

5.3 Domains

We examine the effectiveness of MIVAT using a number of different sources of poker hands. In 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) and for three-player nine-bots from the 2009 AAAI Computer Poker Competition. 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.

We tested for the statistical significance of differences between the variances of the estimators using the Brown-Forsythe test. For two sample variances $\hat{\sigma}_1^2, \hat{\sigma}_2^2$ of sample sizes N_1, N_2, the null hypothesis is $H_0 : \sigma_1^2 = \sigma_2^2$ and the alternate hypothesis $H_a : \sigma_1^2 \neq \sigma_2^2$. If the test statistic $W > F_\alpha(2, (N_1 + N_2) - 2)$, the F-table for a given α, then we can say that the difference is statistically significant with $1 - \alpha$ confidence. We used $\alpha = 0.05$ and F-table[2] with degrees of freedom up to 1000, making the critical value = 1.04.

The test statistic W is defined as

$$W = \frac{(N-k)}{(k-1)} \frac{\sum_{i=1}^{k} N_i (Z_{i\cdot} - Z_{\cdot\cdot})^2}{\sum_{i=1}^{k} \sum_{j=1}^{N_i} (Z_{ij} - Z_{i\cdot})^2}$$

where

- k is the number of different groups of samples (for our comparison, $k = 2$)
- $N = N_1 + N_2$
- S_{ij} is the skill value for the jth sample from the ith group
- $Z_{ij} = |S_{ij} - \tilde{S}_{i\cdot}|$, where $\tilde{S}_{i\cdot}$ is the median of the ith group
- $Z_{i\cdot} = \frac{1}{N_i} \sum_{j=1}^{N_i} Z_{ij}$, the mean of Z_{ij} for group i
- $Z_{\cdot\cdot} = \frac{1}{N} \sum_{i=1}^{k} \sum_{j=1}^{N_i} Z_{ij}$, the mean of all Z_{ij}

[2] http://www.ma.utexas.edu/users/davis/375/popecol/tables/f005.html

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 5.1: 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
Bots-Bots	2.461	2.418	2.506	5.438

Table 5.2: Standard deviation of tailored estimators for two-player, limit poker.

Note that the Brown-Forsythe test with the median is used because it is robust to many types of non-normal data (unlike the F-test) [Brown and Forsythe, 1974]. Other options include using the mean or trimmed mean as opposed to the median, given more knowledge about the underlying distribution.

5.3.1 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 an equal mixture of Bot-Humans and Bot-Bots 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 5.1. First, 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 both the Bots-Bots and Bots-Humans data.

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 5.2 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 5.2. Using the specific function, we now statistically significantly outperform DIVAT on both datasets without using the DIVAT feature. 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.

Data	MIVAT	Money
Bots-Bots	31.354078	44.239889

Table 5.3: Standard deviation of estimators for two-player, no-limit poker. All three-wise products of features were used.

Data	MIVAT	Money
Bots-Bots	2.830141	4.489903

Table 5.4: Standard deviation of estimators for three-player, limit poker.

5.3.2 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. Table 5.3 shows the resulting standard deviation on test data from the AAAI no-limit competition. MIVAT resulted in a 29% 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 further splits on the potsize may result in additional improvements. We might also obtain better splits on the pot-size with a more systematic search or by generating many randomly chosen splits, rather than manual exploration.

5.3.3 Three-Player Limit Poker

As in two-player, no-limit poker, there is no known reduced-variance estimator in three-player, limit poker. We obtain considerable variance reduction using a split on the number of players left in the game. A separate estimator is learned for when there are three or two players in the game, with pot-equity added as a feature when only two players are left.

Table 5.4 shows a variance reduction of about 40% using MIVAT over the Money. Because it is a limit game, this amount of variance reduction is expected. In particular, in many scenarios, one player folds, enabling the use of pot-equity as a feature. We expect that variance reduction comparable to two-player limit is possible with betting splits. For three-player limit poker, there would be more betting scenarios than with two-player limit. If these scenarios could be grouped into a relatively small number of categories, however, the betting splits would likely produce a lower-variance estimator by allowing value functions to be learned on lower-variance segments of the data.

Data	MIVAT	Money
Bots-Bots	4.201377	5.601836

Table 5.5: Standard deviation of estimators for six-player, limit poker.

5.3.4 Six-Player Limit Poker

We expect that even in limit poker, moving to a larger number of players is going to introduce variance. With six-player limit poker, again having no known reduced variance estimator, we do not obtain the same wins as three-player limit poker. MIVAT was trained with the same player split as in three-player, limit poker: separate functions are learned depending on the number of non-folded players. The standard deviation of MIVAT and the Money estimators are shown in Table 5.5. MIVAT shows only a modest 25% variance reduction. This result is somewhat surprising, as the variance of the money is similar to both two-player and three-player limit poker and much lower than the variance in no-limit poker.

The inability to match the success in two-player or three-player limit poker is likely due to the limited feature set for the majority of scenarios (no pot-equity). Similarly to two-player limit, we could learn separate functions for different betting scenarios. Because the amount of players makes enumerating the set of betting strings much more computationally intense, we could look at splits rating player aggression, such as the number of bets or raises from the player. Another option is to explore using pairwise pot-equity for the players, adding more information than hand-strength alone. 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.

5.3.5 Two-Player Limit Poker with Limited Information

The University of Alberta expert-level poker bot, Polaris, which competed in the Second Man-vs-Machine Poker Championship, selects between five different strategies depending on their opponent: passive to aggressive. Currently, they are using a biased version of DIVAT to judge their opponents play. This estimator uses the DIVAT estimate when the game ends in a showdown and all the cards seen; else, it uses the winnings. In practice, the amount of variance reduction overcomes the bias introduced; the gain, however, is small [Bowling *et al.*, 2008].

To obtain an unbiased estimator, we can learn a lower-variance estimator using only one player's cards, the player we are evaluating. This estimator can be used during play when the opponent's cards are not revealed, i.e., when play is terminated due to a fold. In this case, we can only use HS and HS2, as pot-equity relies on knowledge of both hands. With just these simple features and the betting split option using the last player to raise, we obtain a variance reduction of 18% (about 1 small bet). This amount of variance reduction makes it worthwhile for the University of Alberta Poker Group to incorporate the MIVAT estimator into their two player limit poker bot, Polaris.

Data	MIVAT	Money
Bot-Humans	4.607391	5.608046

Table 5.6: Standard deviation of estimators for limited-information two-player, limit poker on data of Polaris playing a variety of human players.

In this chapter, we have seen the practical usefulness of MIVAT in Texas hold'em poker. In the next two Chapters, we show the optimality and the general applicability of MIVAT.

Chapter 6

Optimality of Approach

In the previous chapters, we developed a general framework to facilitate construction of control variates and illustrated its merit in Texas hold'em poker. In this chapter, we prove the optimality of our approach for selecting control variates in extensive games. First, we rest ate the optimality results from the advantage sum work in finite extensive games. Then, we extend these results to the continuous action case when using linear value functions. Finally, we illustrate the optimality of MIVAT when approximating the value function. In particular, we demonstrate the importance of using the MIVAT optimization as opposed to the optimization proposed in the advantage sum work. Note that to simplify the notation, we will write V with the implicit assumption that we mean V_j.

6.1 Optimality of Advantage Sum Estimators

Zinkevich et al. [2006] proved that predicting the expected utility, $V(h) = E_\sigma[u|h]$, produces the optimal value function for variance reduction. The following theorems from the work on advantage sum estimators illustrate the optimality.

Definition 6.1.1 *Given a finite extensive game Γ and strategy profile σ, $\hat{u}^*$ is a **minimum variance unbiased estimator for Γ under σ** if $\hat{u}^*$ is an unbiased estimator and for any unbiased estimator for Γ, $\hat{u}$:*

$$\mathrm{Var}_\sigma[\hat{u}^*] \leq \mathrm{Var}_\sigma[\hat{u}]$$

*We will call a value function V **exact** if $\hat{u}_V$ is a minimum variance unbiased estimator for Γ under σ.*

Theorem 6.1.2 *Assume you are given a finite extensive game Γ. Given strategy profile σ, define $V^\sigma : H \to \mathbb{R}$ as $V^\sigma(h) = E_\sigma[u|h]$. Then, the advantage sum estimator $\hat{u}_{V^\sigma}$ is a minimum variance unbiased estimator for Γ under σ.*

If we can exactly represent the exact value function, then the advantage sum estimator is guaranteed to be a minimum variance estimator. If our representation or data are inadequate to learn the

exact value function, there is a relationship between the variance of the approximate estimator and the exact estimator. The next theorem shows that if the approximate value function is close to the exact value function, then correspondingly the variance of the approximate advantage sum estimator is close to the variance of the minimum-variance estimator.

Theorem 6.1.3 *Assume you are given a finite extensive game* Γ. *Given strategy profile* σ *and* $V, V' : H \to \mathbb{R}$, *define* $u_{max} = \max_{z \in Z} (\max(\hat{u}_V(h), \hat{u}_{V'}(h)))$. *It is the case that:*

$$\mathrm{Var}_\sigma[\hat{u}_V] - \mathrm{Var}_\sigma[\hat{u}_{V'}] \leq 4u_{max} \sum_{h \in H} Pr_\sigma[h]|V(h) - V'(h)| \tag{6.1}$$

In the next section, we extend the optimality results of advantage sum estimators to the continuous action setting for linear value functions. In the following two sections, we explore approximations and inadequate representations for the value function, justifying minimizing the variance as opposed to predicting the utility as suggested in the advantage sum analysis.

6.2 Optimality of MIVAT

The optimality result for advantage sum estimators applies only to finite extensive games. In this section, we prove the optimality of the MIVAT framework for the continuous action setting when the exact value function is linear. The more general result on non-linear value functions is left for future work.

Definition 6.2.1 *For an extensive game* Γ *and* $C(H) = \{ha | h \in H, a \in A(h),\ P(h) = c\}$ *(all histories following chance nodes), let* $\mathcal{F} \subseteq [C(H) \to \mathbb{R}]$, *a subset of the set of all possible value functions. We will say that* V *is* ***representable by*** $\mathcal{F}$ *(or implicitly just* ***representable****) if* $V \in \mathcal{F}$.

First, we show that the MIVAT estimator is optimal in finite extensive games for any class of value functions, $\mathcal{F}$.

Theorem 6.2.2 *For a finite extensive game* Γ, *strategy profile* σ *and function class* $\mathcal{F}$, *let* $V^*_{var} \in \mathcal{F}$ *be a value function minimizing Equation 4.5 (i.e. minimizing the variance) and* $V^*_w \in \mathcal{F}$ *with* $V^*_w(h) = E[u|h, \sigma]$ *the advantage sum value function. Then* $\mathrm{Var}(\hat{u}_{V^*_{var}}) = \mathrm{Var}(\hat{u}_{V^*_w})$.

Proof: Assume $\exists\ V^*_w, V^*_{var} \in \mathcal{F}$ such that $\mathrm{Var}(\hat{u}_{V^*_{var}}) > \mathrm{Var}(\hat{u}_{V^*_w})$
$\implies \mathrm{Var}(\hat{u}_{V^*_w}) < \mathrm{Var}(\hat{u}_{V^*_{var}}) = \min_{V \in \mathcal{F}} \mathrm{Var}(\hat{u}_V)$
which is a contradiction. ∎

To prove that the MIVAT estimator is optimal for extensive games, we will assume that for given feature function $\phi : H \to \mathbb{R}^d$, the exact value function is linear,

$$V^* \in \mathcal{F}^\phi_{lin} = \{\theta \in \mathbb{R}^d : V(h) = \phi(h)^T\theta,\ \forall h \in H\}$$

Theorem 6.2.3 *Assume you are given an extensive game Γ and strategy profile σ where $V^* \in \mathcal{F}^{\phi}_{lin}$. The MIVAT estimator $\hat{u}_\theta$ is a minimum variance unbiased estimator for Γ under σ.*

Before proving this theorem, we need to prove that there exists a minimum variance unbiased estimator. Then the MIVAT optimization is well-defined and will find the minimum variance estimator.

First, we will provide some labels for convenience. Given an extensive game Γ, strategy profile σ and feature function $\phi : H \to \mathbb{R}^d$ with $\phi_1 = u$, let

1. $A(z)$ be the feature changes in z, for $z \in Z$

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

2. U_b the set of unbiased estimators with linear value functions, $U_b = \{\hat{u}_\theta = \begin{bmatrix} 1 \\ \theta \end{bmatrix} : \theta \in \mathbb{R}^d\}$.
3. $M = \sup_{z \in Z} u(z)$ (which exists because Z is bounded and u finite)
4. $U_{o,\sigma} = \{\hat{u} \in U_b \ : \ |\hat{u}(z)| \leq M \ \forall z \in Z \text{ and } \mathrm{Var}[\hat{u}|\sigma] \leq \mathrm{Var}[u|\sigma]\}$

We will drop σ for convenience when it is clear that we are talking about a specific σ.

Lemma 6.2.4 *For a given σ, $\mathrm{Var}[\hat{u}_\theta|\sigma]$ is continuous on U_o.*

Proof: Then notice that

$$\begin{aligned}
\mathrm{Var}[\hat{u}_\theta|\sigma] &= \int_Z \Pr_\sigma(z) \left(u(z) - A(z)^T\theta\right)^2 dz - \left(\int_Z \Pr_\sigma(z) \left(u(z) - A(z)^T\theta\right) dz \right)^2 \\
&= \int_Z \Pr_\sigma(z) \left(u^2(z) - u(z)A(z)^T\theta + \theta^T A(z)A(z)^T\theta\right) dz - \left(E[u|\sigma]\right)^2 \\
&= E[u^2|\sigma] - \left(\int_Z \Pr_\sigma(z)u(z)A(z)^T dz \right)\theta + \theta^T \left(\int_Z \Pr_\sigma(z)A(z)A(z)^T \right)\theta - \left(E[u|\sigma]\right)^2 \\
&= \mathrm{Var}[u|\sigma] - \left(\int_Z \Pr_\sigma(z)u(z)A(z)^T dz \right)\theta + \theta^T \left(\int_Z \Pr_\sigma(z)A(z)A(z)^T \right)\theta
\end{aligned}$$

Since $\mathrm{Var}[\cdot|\sigma]$ is quadratic on a subset of a vector space ($U_o \subset \mathbb{R}^d$), it is continuous.

■

Lemma 6.2.5 *Given σ, U_o is a non-empty, compact set.*

Proof: U_o is non-empty because $\hat{u}_\theta \in U_b$ for $\theta = \mathbf{0}$.

Since $U_o \subset \mathbb{R}^d$, we need only show that it is closed (by definition, it is bounded).

Recall that for a continuous function $f : K \to \mathbb{R}$, for a convergent sequence $\{k_i\} \subset K$ with $k_i \to_\infty k$, then $\{f(k_i)\}$ converges to $f(k)$.

Take any convergent sequence $\{\theta_i\} \subset U_o$, with $\theta_i \to \theta$. This means that

$$\hat{u}_{\theta_i} = \begin{bmatrix} 1 \\ \theta_i \end{bmatrix} \to \hat{u}_\theta = \begin{bmatrix} 1 \\ \theta \end{bmatrix}$$

Claim 1: $|\hat{u}_\theta(z)| \leq M \; \forall z \in Z$

Assume that $\exists z \in Z$ such that $|\hat{u}_\theta(z)| > M$. Then there must exist $\epsilon > 0$ such that $|\hat{u}_\theta(z)| = M + \epsilon$. Then we know that there are infinitely many $|\hat{u}_{\theta_i}(z)| \in B_\epsilon(\hat{u}_\theta(z))$

. $\implies \exists n \in \mathbb{N}$ such that $|\hat{u}_{\theta_n}(z)| > M$

which is a contradiction.

Claim 2: $\text{Var}[\hat{u}_\theta|\sigma] \leq \text{Var}[u|\sigma]$

Because the $\text{Var}[\cdot|\sigma]$ is a continuous function, the same argument as in Claim 1 applies.

Therefore, $\hat{u} \in U_o$. ∎

Because we have a continuous function on a non-empty compact set, we can now show that the variance function attains a minimum on U_o. The remaining question, however, is whether U_o contains all the estimators with variance lower than the utility. The following lemma addresses this issue, allowing us to finally conclude that a minimum variance unbiased estimator exists.

Lemma 6.2.6 *Given σ, if $\hat{u} \notin U_o$ then $\exists \hat{u}' \in U_o$ with* $\text{Var}[\hat{u}'|\sigma] \leq \text{Var}[\hat{u}|\sigma]$.

Proof: Let $Z' = \{z \in Z : \hat{u}(z) > M\}$.

If $m(Z') = 0$, then we can set $\hat{u}(Z') = M$, leaving the mean and variance unchanged but now obtaining $\hat{u} \in U_o$.

Else, let $B = \int_{Z'} \Pr_\sigma(z)\hat{u}(z)dz$ and $\mu = E[u|\sigma]$. Let

$$\hat{u}_1(z) = \begin{cases} M & \text{if } z \in Z' \\ \hat{u}(z) & \text{if } z \notin Z' \end{cases}$$

giving

$$\int_{Z'} \Pr_\sigma(z)\hat{u}_1(z)dz = m(Z')M$$

The estimator $\hat{u}_1$ now satisfies $\hat{u}_1(z) \leq M \; \forall z \in Z$, but is biased. To make $\hat{u}_1$ unbiased, we will distribute $B - m(Z')M$ amongst $Z_\mu = \{z \in Z \backslash Z' : \hat{u}(z) \leq \mu\}$. If we assume that

$$m(Z_\mu)\mu - \int_{Z_\mu} \Pr_\sigma(z)\hat{u}(z)dz \geq B - m(Z')M$$

(which we prove later), then $\exists Z'' \subset Z_\mu$ with

$$\int_{Z''} \Pr_\sigma(z)\hat{u}(z)dz = m(Z'')\mu - B + m(Z')M$$

Set $\hat{u}_1(Z'') = \mu$. This means that

$$\begin{aligned}
E[\hat{u}_1|\sigma] &= m(Z')M + m(Z'')\mu + \int_{Z\setminus(Z'UZ'')} \Pr_{\sigma}(z)\hat{u}(z)dz \\
&= B + m(Z')M + m(Z'')\mu - B + \int_{Z\setminus(Z'UZ'')} \Pr_{\sigma}(z)\hat{u}(z)dz \\
&= B + \int_{Z\setminus Z'} \Pr_{\sigma}(z)\hat{u}(z)dz \\
&= E[\hat{u}|\sigma]
\end{aligned}$$

This means that $\hat{u}_1$ is now an unbiased estimator. Notice that values above and below μ were shifted closer to μ for $\hat{u}_1$. Consequently, $\hat{u}_1$ has lower variance than $\hat{u}$.

The remaining problem is that we assumed

$$m(Z_\mu)\mu - \int_{Z_\mu} \Pr_{\sigma}(z)\hat{u}(z)dz \geq B - m(Z')M$$

Assume that $m(Z_\mu)\mu - \int_{Z_\mu} \Pr_{\sigma}(z)\hat{u}(z)dz < B - m(Z')M$. Then

$$\begin{aligned}
E[\hat{u}|\sigma] &= \int_{Z'} \Pr_{\sigma}(z)\hat{u}(z)dz + \int_{Z_\mu} \Pr_{\sigma}(z)\hat{u}(z)dz + \int_{Z\setminus(Z'UZ_\mu)} \Pr_{\sigma}(z)\hat{u}(z)dz \\
&\geq B + \mu m(Z_\mu) - B + m(Z')M + (1 - m(Z') - m(Z_\mu))\mu \\
&= m(Z')M + (1 - m(Z'))\mu \\
&> \mu
\end{aligned}$$

which is a contradiction.

■

In this section, we have illustrated that the MIVAT framework is optimal for finite extensive games in continuous action settings when the exact value function is linear. In the next section, we demonstrate the importance of the MIVAT optimization when approximating the value function from data or when using inadequate representations for the learned value functions.

6.3 Significance of the MIVAT Loss Function

The question remains as to the importance of choosing the MIVAT optimization over predicting the expected utility, as both optimizations produce a minimum-variance estimator. The difference between these optimizations becomes important when we approximate the exact value function. Intuitively, minimizing the variance (our ultimate goal) is the appropriate approach for obtaining a minimum variance approximation, as opposed to the indirect approach of predicting the utility. In the previous sections, we have seen that $\mathrm{Var}[\hat{u}_{V_{var}}|\sigma] \leq \mathrm{Var}[\hat{u}_{V_w}|\sigma]$. If the two estimators have different variance, then $Var(\hat{u}_{V_{var}}) < Var(\hat{u}_{V_w})$. As the two loss functions optimize different criteria, we would often expect that the variance be different when approximating the value function.

This section formalizes this intuition. First, we prove a data-dependent result illustrating when MIVAT is more likely to reduce variance on testing data. Then, we provide an example showing that minimizing the variance produces a lower-variance estimate than predicting the expected utility. Finally, we further verify the importance of using the MIVAT optimization by comparing empirical results on two poker domains: two-player limit and two-player no-limit poker.

6.3.1 Significance of the Loss Function for Learning Approximations

In this section, we prove a data-dependent result showing scenarios where the MIVAT estimator is more accurate than the advantage sum estimator. To compare the two approaches, we need to define an optimization for predicting the expected utility. A common optimization is the L_2 loss: the mean-squared error between the value function and the utility. We will show later that the choice of L_2 loss was not pivotal for the comparison. With the L_2 loss, we obtain the following optimization:

$$\underset{V}{\textbf{Minimize:}} \sum_{t=1}^{T} \left(\sum_{\substack{ha \sqsubseteq z_t \\ P(h)=c}} \left(V(ha) - u(z_t)\right)^2 \right) \tag{6.2}$$

In the following proof, we will illustrate the relationship between the variance of the MIVAT estimator and the predict utility estimator. Interestingly, we obtain a data-dependent result illustrating that, with enough training data and a substantial difference in the training losses, the superiority of the MIVAT estimator on the training data extends to test data.

Before beginning the proof, we will need the following definitions.

1. Let X_T be the training data with T samples and d columns.

2. Let the MIVAT loss function on X_T be

$$\hat{L}_{var}(\hat{u}_V) = \sum_{t=1}^{T} \left[(\hat{u}_V(z_t) - \sum_{t=1}^{T} \hat{u}_V(z_t) \right]^2 .$$

3. Let X be the testing data with n samples and d columns.

4. Let the MIVAT loss function on X be

$$L_{var}(\hat{u}_V) = \sum_{t=1}^{n} \left[(\hat{u}_V(z_t) - \sum_{t=1}^{n} \hat{u}_V(z_t) \right]^2 .$$

Definition 6.3.1 *The **Rademacher complexity** of a class of real-valued functions $\mathcal{F}$ for a sample $z = (z_1, \ldots, z_d) \in Z$ of size d is*

$$R_d(\mathcal{F}) = E[\hat{R}_z(\mathcal{F})]$$

where $\hat{R}_z(\mathcal{F})$ is the empirical Rademacher complexity of $\mathcal{F}$

$$R_z(\mathcal{F}) = \frac{2}{d} E[\sup_{f \in \mathcal{F}} \left| \sum_{i=1}^{d} \sigma_i f(z_i) \right|$$

The independent random variables $\sigma_1, \ldots, \sigma_d$ satisfy $\Pr(\sigma_i = 1) = \Pr(\sigma_i = -1) = \frac{1}{2}$ for any $i = 1, \ldots, d$.

Definition 6.3.2 *The **Lipschitz constant** of a function $f : X \to Y$ of two metric spaces $(X, d_X), (Y, d_Y)$ is the smallest $K \geq 0$ such that*

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2)$$

We will call the Lipschitz constant of a function $K(f)$. Notice that $\mathrm{Var}[\cdot|\sigma] : \mathcal{F} \to \mathbb{R}$ for $\mathcal{F}$ a bounded set has a finite Lipschitz constant.

Theorem 6.3.3 *Given any extensive form game Γ, any $\delta \in (0, 0.5)$ and classes of bounded functions $\mathcal{F}$ with bounded Rademacher complexity $R_d(\mathcal{F})$, then for a given X_T if*

$$\hat{L}_{var}(\hat{u}_{V_w}) - \hat{L}_{var}(\hat{u}_{V_{var}}) \geq 4K(L_{var})E[R_d(\mathcal{F}(X_T))] + B\sqrt{\frac{2log(\frac{2}{\delta})}{T}}$$

for constant B, then

$$\Pr(L_{var}(\hat{u}_{V_{var}}, X) < L_{var}(\hat{u}_{V_w}, X)) > 1 - \delta.$$

Proof: Let $L = L_{var}$ for ease of notation. Notice that we can compute $\gamma = \hat{L}(\hat{u}_{V_w}) - \hat{L}(\hat{u}_{V_{var}})$ empirically, with $\gamma > 0$.

From Boucheron et al. (Section 4.1)[2005], we can see that with probability $\frac{\delta}{2}$

$$\begin{aligned}
L(\hat{u}_{V_{var}}) &\leq \hat{L}(\hat{u}_{V_{var}}) + \sup_{V \in \mathcal{F}} (L(u_V) - \hat{L}(u_V)) \\
&\leq \hat{L}(\hat{u}_{V_w}) - \gamma + \sup_{V \in \mathcal{F}} (L(u_V) - \hat{L}(u_V)) \\
&\leq \hat{L}(\hat{u}_{V_w}) - \gamma + 2E[R_d(L \circ \mathcal{F}(X_T))] + B_1\sqrt{\frac{2log(\frac{2}{\delta})}{T}} \\
&\leq \hat{L}(\hat{u}_{V_w}) - \gamma + 2K(L)E[R_d(\mathcal{F}(X_T))] + B_1\sqrt{\frac{2log(\frac{2}{\delta})}{T}}
\end{aligned}$$

where $L \circ \mathcal{F}$ is composition. Refer to Boucheron et al. work [2005] for a proof of these steps. Similarly, with probability $\frac{\delta}{2}$

$$\begin{aligned}
\hat{L}(\hat{u}_{V_w}) &\leq L(\hat{u}_{V_w}) + \sup_{V \in \mathcal{F}} (\hat{L}(u_V) - L(u_V)) \\
&\leq L(\hat{u}_{V_w}) + 2K(L)E[R_d(\mathcal{F}(X_T))] + B_2\sqrt{\frac{2log(\frac{2}{\delta})}{T}}
\end{aligned}$$

The event spaces not satisfying these two inequalities may be disjoint, but each has at most probability $\frac{\delta}{2}$. Therefore, we can combine the two inequalities (with $B = B_1 + B_2$)

$$\Pr\left(L(u_{V_{var}}) \leq L(\hat{u}_{V_w}) - \gamma + 4K(L)E[R_d(\mathcal{F}(X_T))] + B\sqrt{\frac{2log(\frac{2}{\delta})}{T}} \right) \geq 1 - \delta$$

Now we can see that if

$$\gamma \geq 4K(L)E[R_d(\mathcal{F}(X_T))] + B\sqrt{\frac{2log(\frac{2}{\delta})}{T}}$$

then

$$\Pr\left(L(\hat{u}_{V_{var}}) \leq L(\hat{u}_{V_w}) \right) \geq 1 - \delta$$

■

Interestingly, in the above proof, choosing the estimator u_{V_w} was irrelevant. For any loss function $L \neq L_{var}$ producing a bounded estimator $\hat{u}_V$ with bounded Rademacher complexity, we would find $\gamma > 0$. The superiority of the MIVAT estimator, however, will differ over the u_V and X_T; the γ difference will vary from significant to negligible.

6.3.2 Significance of the Loss Function with Inadequate Representations

Another important factor when approximating the exact value function is the *representation*. For example, if the exact value function is non-linear and we are using linear value functions, then our representation is inadequate. We have already seen that when approximating the value function, $\hat{u}_{\theta_{var}}$ is likely to have lower variance than $\hat{u}_{\theta_w}$. Similarly, with an inadequate representation for the value function, we would expect the variance to be different, causing $\mathrm{Var}[\hat{u}_{V_{var}}|\sigma] < \mathrm{Var}[\hat{u}_{V_w}|\sigma]$. The below theorem provides an example of this phenomena when using a linear value function to represent a non-linear exact value function.

Theorem 6.3.4 *There exists a game* Γ *where for any* σ, $\mathrm{Var}[\hat{u}_{\theta_{var}}|\sigma] < \mathrm{Var}[\hat{u}_{\theta_w}|\sigma]$.

Proof: Lets look at the following single player game and assume that V_{var} and V_w are linear. The game begins by dealing the player one card from the two-card deck, {K, Q}. If the player is dealt a K, then he wins 4; else he loses 1. The features are 2 if a K is dealt and -1 if a Q is dealt.

Since we can only use a linear weighting of the features, we will not be able to predict the utility: $u = \phi|\phi|$. The two samples where P1 is dealt a Q on the first game and a K on the second game encapsulate all possible hands. Because the {K, Q} are dealt uniformly,

$$\sum_{a' \in A(h)} f_c(a'|h)\phi(ha') = 0.5 \cdot (-1) + 0.5 \cdot 2 = 0.5$$

Below are the u_t, A_t values for the two game samples:

$$u_1 = -1, \quad \phi_1 = -1, \quad A_1 = \phi_1 - \tfrac{1}{2} = -\tfrac{3}{2}$$
$$u_2 = 4, \quad \phi_2 = 2, \qquad A_2 = \phi_2 - \tfrac{1}{2} = \tfrac{3}{2}$$
$$A = 0, \quad \overline{u} = \tfrac{1}{2}(-1+4) = \tfrac{3}{2}$$

We can compute θ_{var} by substituting these values into the MIVAT estimator solution (Equation 4.8):

$$\begin{aligned}\theta_{var} &= \left(\sum_{t=1}^{2} \frac{A_t^2}{2}\right)^{-1} \left(\sum_{t=1}^{2} \frac{A_t u_t}{2}\right) \\ &= \frac{5}{3}\end{aligned}$$

This gives us skill computations for the two games

$$u_1^{\theta_{var}} = -1 + \frac{3}{2} * \frac{5}{3} = 1.5$$
$$u_2^{\theta_{var}} = 4 - \frac{3}{2} * \frac{5}{3} = 1.5$$

with a variance of 0.

Similarly, we can solve for θ_w

$$\begin{aligned}\frac{\partial C(\theta_w)}{\partial \theta_w} &= \sum_{t=1}^{2} p(u_t - \phi_t \theta_w)^{p-1}(-\phi_t) \\ &= (-1+\theta_w)^{p-1}(-1) + 2(4-2\theta_w)^{p-1} \\ &= 2^p(2-\theta_w)^{p-1} - (-1+\theta_w)^{p-1} \qquad \text{and } p = 2 \\ &= 4(2-\theta_w) - (-1+\theta_w) \\ &\implies \theta_w = \frac{9}{5}\end{aligned}$$

The corresponding skill calculations are

$$u_1^{\theta_w} = -1 + \frac{3}{2} * \frac{9}{5} = 1.7$$
$$u_2^{\theta_w} = 4 - \frac{3}{2} * \frac{9}{5} = 1.3$$

with a variance of 0.08, as opposed to a variance of 0 for the MIVAT estimator. Because it was not possible to predict the utility, using the advantage sum optimization needlessly introduced variance.

■

The remaining question is the importance of using the L_2 loss. In the above proof, we illustrated that minimizing the variance outperforms minimizing the L_2 loss. But what about other L_p norms, which do not have a closed form solution for general problems? One interesting thing to notice in the above example is that using the L_1 or L_∞ norm instead of L_2 gives the same solution as θ_{var}.

Poker Domain	MIVAT	Predict Winnings	Money
2-Player Limit	2.108126	2.348987	5.608015
2-Player No-Limit	31.354078	32.909832	44.239889

Table 6.1: Comparison of the standard deviation of small bets for two estimators in two poker domains.The comparison is obtained from the same train-test splits and feature choices described in the results section. For two-player limit poker, the DIVAT feature was not used.

Moreover, for increasing $p > 2$ for the L_p loss, θ_w converges to θ_{var}. This phenomena, however, only occurs when there is one feature; with more than one feature, this equivalence will not occur generally.

In a more complicated game not included in the proof for computational clarity, we see that the solutions using the L_p norm $\forall\, p$ are different from θ_{var}. The game involves two deals of $\{1, 2, 3, 4\}$ without replacement. If the sum of the player's two cards is higher than the sum of the two undealt cards, then the player receives $u = 4$; else, he receives $u = -1$.

We have seen that when approximating the value function, the MIVAT optimization is the principled approach. We complete the discussion on the optimality of MIVAT by comparing minimizing the variance and predicting the utility in a real application: Texas hold'em poker.

6.3.3 Empirical Evidence

We expect that MIVAT would be an improvement over predicting the expected utility in general when learning the value function. We compare two Texas hold'em poker domains to illustrate differences on real-world problems in Table 6.1. For both poker domains, minimizing the variance on the training set results in a statistically significant lower-variance estimate than predicting the utility. Because predicting the expected utility is proven to obtain the minimum variance estimator in ideal settings, we see that it is still comparable to MIVAT.

We have seen from this section, that in theoretical and practical scenarios, MIVAT is the optimal approach for obtaining unbiased, low-variance performance estimates using control variates. In the next Chapter, we discuss the broad applicability of the MIVAT framework, making it both an optimal and a general approach.

Chapter 7

Generality of Extensive Games and the MIVAT Framework

We have seen that MIVAT is an optimal approach to facilitating the use of control variates, but how applicable is the framework? The term *extensive game* may make the scope of the domain class unclear. Moreover, it may seem difficult in practice to provide all constituents in the extensive game definition.

In this Chapter, we discuss these issues and illustrate the generality of MIVAT. We first discuss the many domains that fit into the formalism. As a representative example, we show that the widely used domain class, finite-horizon (partially-observable) Markov decision processes, are extensive games. In the following section, we discuss the three major assumptions in our domain class: (1) the requirement for a finite number of interactions (finite-horizon); (2) the knowledge of the chance node dynamics; and (3) the discrete-time assumption. Though discrete time and knowledge of the dynamics are common assumptions, relaxing these assumptions may be important for practical scenarios. We demonstrate relaxations to these assumptions, further illustrating the broad applicability of the MIVAT framework.

7.1 Scope of Extensive Games

Extensive games represent sequential decision making domains, common in real-world scenarios. Consequently, many prevailing domain models fit into our formalism, corresponding to three common scenarios. The first is the single agent scenario usually modeled by finite-horizon (partially-observable) Markov decision processes (MDPs and POMDPs). This scenario is useful for tasks such as learning controllers or option pricing agents. Second, the multi-agent scenario generalizes beyond single-agent, incorporating N-player general and constant-sum games such as poker, trading agents and multi-robot interaction. The final generalization is to the continuous action settings, useful for applications such as robotic arms for grasping, self-balancing robots and autonomous vehicles. In this section, we illustrate that finite-horizon MDPs and POMDPs are extensive games. The goal

is to illustrate, with MDPs as a representative domain class, that seemingly complex, widely-used domains naturally fit into the extensive game formalism.

7.1.1 Finite-horizon MDPs and POMDPs

MDPs and POMDPs are a general model of decision making tasks and have been extensively used and studied [Puterman, 1994]. Many domains can be modeled as MDPs, including real-world tasks such as inventory management, highway pavement maintenance and communication models (for a more detailed discussion and more examples, see [Puterman, 1994]. Notable successes using MDPs include elevator control superior to the state-of-the-art [Crites and Barto, 1996] and autonomous helicopter control that can perform maneuvers beyond a human controller's capabilities [Ng *et al.*, 2004].

An MDP models the relationship between an agent and its environment, formally represented by the tuple (S, A_p, Pr, R) where:

1. S is a finite set of **states**
2. A_p a finite set of **actions**
3. T a **probability function** giving the probability, $T(s'|s, a)$, of transitioning to state s' after taking action a in state s and
4. $R : S \times A_p \times S \to \mathbb{R}$ is a **reward function**.

The term *Markov* means that the current state is only dependent upon the previous state: the transition probabilities are governed only by the previous state and action, $T(s'|s, a)$. Starting from some initial state s_0 determined by $T(s_0|\emptyset)$, each time step the agent is in some state $s \in S$, takes action $a \in A_p$, transitions to state s' with probability $T(s'|s, a)$ and receives reward $R(s, a, s')$. On each step, to enable appropriate decision-making, the agent observes the current state of the environment. *Finite-horizon* MDPs require terminal states (such as goals); the agent must reach a terminal state in n or less steps.

Note that the majority of MDP literature uses two types of MDPs: finite and infinite-horizon MDPs [Puterman, 1994]. The reinforcement learning community defines three types: finite, indefinite and infinite-horizon MDPs. In *indefinite-horizon* MDPs, interaction can be arbitrarily long but must eventually terminate [Sutton and Barto, 1998]. MIVAT still applies to indefinite-horizon MDPs because there is a finite length of interaction. Moreover, in practice, the length of interaction can always be bounded by some n to allow for the idea of indefinite interaction. Finite-horizon and indefinite-horizon are almost equivalent, as neither make assumptions about the agents knowledge of the length of interaction and we can make n large (as the agent only needs to terminate at some time before n steps). For this reason and because the majority of the MDP community uses finite-horizon MDPs, we will focus on the finite-horizon form.

In a partially observable MDP (POMDP), the agent cannot directly observe the state. Instead, it receives an observation representing the state. A POMDP includes a finite set of observations O and probabilities of seeing an observation in a state s' after taking action a: $Pr(o|s', a)$. This formalism is a general one for modeling real-world scenarios, but is much more computationally complex than MDPs.

It is not immediately clear that finite-horizon MDPs and POMDPs are extensive games. In the following, we explicitly describe the relationship.

Fact 7.1.1 *Finite-horizon MDPs are extensive games.*

Proof: Define the equivalent extensive game as follows:

1. Set the number of players, $N = 1$

2. Let $A_c = \{c_1, \ldots, c_{|S|}\}$ be chances actions where c_i corresponds to choose state $s_i \in S = \{s_1, \ldots, s_{|S|}\}$. The extensive game action set A is the union of the agent's actions A_p and chances actions A_c.

3. The histories, H, is a finite set of possible agent-chance action sequences.

4. Set the player function $P(\emptyset) = c$ (chance acts first, choosing the start state s_0) and else to alternate between chance and the agent:

$$P(h) = \begin{cases} 1 & \text{if } mod(|h|, 2) \neq 0 \\ c & \text{if } mod(|h|, 2) = 0 \end{cases}$$

5. For a given history $h = c_{i_1} a_{i_1}, \ldots, c_{i_{i_n}} a_{i_n}$, for all possible $c_{i_{n+1}} \in A(h)$, the Markov property gives the probability measure:

$$f_c(c_{i_{n+1}}|h) = T(s_{i_{n+1}}|h) = T(s_{i_{n+1}}|s_{i_n}, a_{i_n})$$

6. There is complete observability, so the information sets are simply the histories:

$$\mathbf{I}_1 = \{\{h\} : h \in H, P(h) = 1\}$$

7. The utility function $u_1(z)$ is the *return*, the cumulative reward until reaching some terminal state s_m, $m \in \mathbb{N}$:

$$u_1(z) = R_0(z) = \sum_{j=1}^{m-1} R(s_{i_j}, a_{i_j}, s_{i_{j+1}}).$$

   For many trials T, we want to estimate the expected return of the agent: $E[R_0(z)|\sigma]$[1].

■

[1] To unify notation, σ for MDPs is usually called the *policy*, π

Fact 7.1.2 *Finite-horizon POMDPs are extensive games.*

Proof: The justification for Fact 7.1.2 is similar to above, but now we need to redefine the probability measures to incorporate the observations probabilities.

$$f_c(o|ha) = \sum_s Pr(o|s,a)Pr(s|h)$$

where, $Pr(s|h)$ can be calculated recursively.

$$Pr(s|h'ao) = \sum_{s'} Pr(s'|h')Pr(o|h',a)$$

$Pr(s|\emptyset) =$ initial belief distribution (which is required in a finite horizon framework).

■

The remaining problem is that the restriction to an finite-horizon excludes the class incorporating continuing interaction. *Infinite-horizon* MDPs and POMDPs are also widely used and in the next section (Section 7.2.1), we illustrate how infinite-horizons scenarios can also be evaluated in our framework.

7.2 Relaxations of Assumptions

The three major assumptions in our domain class include the requirement for a finite number of interactions; the knowledge of the chance node dynamics; and the discrete-time assumption. With certain modifications, however, we can extend the applicability to scenarios without these assumptions. This section describes these relaxations and completes the discussion on the generality of our framework.

7.2.1 Infinite-horizon Extension

MIVAT, as presented, can only be applied to scenarios with a finite number of interactions; a utility on terminal histories is required for the (Monte Carlo) estimator to be well-defined. Certain scenarios, however, are more naturally formulated assuming interaction continues infinitely without guaranteed termination. For example, a robot with a long life-span would be formulated as a continuing task [Sutton and Barto, 1998]. We illustrate how MIVAT can be applied to infinite-horizon MDPs to relax the finite-interactions requirement.

In infinite-horizon MDPs, the agent-environment interaction cannot be broken down into identifiable episodes, but continues indefinitely. Because the number of steps $n = \infty$, the return $R_n = \sum_{k=n}^{\infty} r_{k+1}$ could be infinite. Most commonly, returns are extended to the continuous case by *discounting* rewards into the future. This definition, however, is required for defining values of *states*. We are devising a metric for the *agent*. An accurate reflection of the agent's skill is $E[R|\sigma]$,

the expected average reward for policy σ in the MDP. Below we show that average expected reward is well-defined and enables evaluation using MIVAT for infinite-horizon MDPs without using discounting.

To make evaluation unbiased, we need to assume that the infinite-horizon MDP is *ergodic*. First, we define ergodicity and then illustrate that average reward is a suitable utility for infinite-horizon MDPs.

Definition 7.2.1 *[Neal, 1993] For a Markov chain $\{S_t\}$ with (time-homogenous) transition probabilities $T : S \times S$, $p_t(s)$ is the probability of a state $s \in S$ occurring at time t, computed using the transition probabilities*

$$p_t(s) = \sum_{s' \in S} p_{t-1}(s')T(s', s)$$

Given the initial probabilities $p_0 : S \to [0, 1]$, p_t determines the behaviour of the chain.

*A distribution over the states of a Markov chain, ρ, is **invariant** if*

$$\rho(s) = \sum_{s' \in S} \rho(s')T(s', s)$$

A Markov chain is ergodic if

1. *there exists an invariant distribution ρ*
2. *$p_t(s)$, converges to this invariant distribution as $t \to \infty$ regardless of the choice of $p_0(s)$*

Intuitively, an ergodic Markov chain does not allow transitions into a part of the state space without some probability of transitioning out. More formally, the probabilities $p_t(s)$ cannot converge to a cycle of distributions. For example, a state space with the probability of transitioning into two separate games (each unreachable from the other) is not ergodic because there are two invariant distributions dependent on the choice of $p_0(s)$. A large class of Markov chains are ergodic [Neal, 1993], making ergodicity a reasonable assumption. Notice that we are using time time-homogenous transition probabilities: T does not depend on time t.

Definition 7.2.2 *An infinite-horizon MDP is ergodic if for each policy σ, the Markov chain induced by σ is ergodic.*

Fact 7.2.3 *An ergodic infinite-horizon MDP can be evaluated under the MIVAT framework as an extensive game with $H = S$ (i.e., episode length 1).*

Proof: Assume we are given a trajectory, $h = c_{i_1}, a_{i_1}, \ldots, c_{i_n}, a_{i_n}, c_{i_{n+1}}$, generated by σ and (time-homogenous) transition probabilities T, where $i_j \in \{1, ..., |S|\}$ and n can be ∞. Define the utility for each state s_{i_j} (corresponding to chance action c_{i_j}):

$$u(s_{i_j}) = R(s_{i_j}, a_{i_j}, s_{i_{j+1}})$$

Because of the ergodicity assumption, we know that [Ortner, 2007]

1. for a finite n, $\frac{1}{n}\sum_{j=1}^{n-1} u(s_{i_j})$ is an unbiased estimate of $E[R|\sigma]$

2. $\lim_{n\to\infty} \frac{1}{n}\sum_{j=1}^{n-1} u(s_{i_j}) = E[R|\sigma]$.

The length of an episode is one with the horizon now representing the number of samples, allowing $n \to \infty$. The MIVAT estimator removes luck on each time step, giving

$$\hat{u}_{V_1}(s_{i_j}) = u(s_{i_j}) - \left[V_1(s_{i_j}) - \sum_{s'\in S} T(s_{i_j}, a_{i_j}, s')V_1(s')\right]$$

which is unbiased because

$$\begin{aligned} E\left[V_1(s_{i_j})\right] &= \sum_{s\in S}\sum_{a\in A}\sum_{s'\in S} \sigma(a|s)T(s,a,s')V_1(s') \\ &= E\left[\sum_{s'\in S} T(s_{i_j}, a_{i_j}, s')V_1(s')\right] \end{aligned}$$

■

In practice, agent interaction with the environment is often terminated after some maximum number of steps M: the performance measure is the cumulative reward[2] from the M steps averaged over t runs (for more significant results). Consequently, we can evaluate infinite-horizon POMDPs similarly to finite-horizon POMDPs: define the utility to be the cumulative reward over M steps. Notice that this metric is equivalent to average reward, where average reward is the cumulative reward scaled by $\frac{1}{M}$. The magnitude is shifted but the ordering of skill remains unchanged.

7.2.2 Approximating the Dynamics

In situations where explicitly specifying the probability measure f_c is infeasible, we can use approximations. There are two options to approximating the dynamics: (1) providing a prior (a guess of the distribution function) or (2) estimating the dynamics using frequency counts from sampling. The first approach could introduce bias, due to an incorrect probability measure. If the magnitude of the bias is less than the magnitude of variance reduction (making the mean-squared error decrease), this approach may be desirable.

The second approach is unbiased but the accuracy dependent on the number of samples. In simulated scenarios, the simulator can be repeatedly sampled to estimate the probability measures. In real-world applications, a limited number of samples could lead to inaccuracies; variance reduction techniques like importance sampling could be used to improve the approximation. Overall, with a reasonably accurate estimate of the dynamics, we anticipate that the amount of variance reduction will overcome any bias or inaccuracies introduced by the two approaches respectively.

[2] Average reward is also a common metric, which is what we defined the utility to be above.

7.2.3 Continuous Time

Finally, we can extend this formalism to include continuous time.[3] For finitely many discrete actions and continuous time, we show how to extend MIVAT by using the finite action space to discretize the time dimension. For continuous actions and continuous time, agents and chance act concurrently; it is not clear how to separate the influence of chance on the utility. For this case, we can only look at situations without agents, i.e., stochastic processes. This scenario is still useful for obtaining low variance estimates of the expected value of stochastic processes, such as for option pricing.

We will provide an example to make the extension for finite discrete actions intuitively clear. Consider a basketball shoot off between two teams of players where the first team to make twenty shots is the winner. Team one has t_1 players each with probability p_1 of making a shot and team two has t_2 players each with probability p_2 of making a shot. Let $s_1(t), s_2(t)$ be the number of shots sunk by teams one and two respectively by time t. Then the time dimension can be discretized by looking at the transitions into states $\{(s_1(t), s_2(t))\} = \{(0,0), (0,1), \ldots, (0,20), (1,0), \ldots, (20,19)\}$. The terminal states are

$$Z = \{z = (s_1^1, s_2^1), (s_1^2, s_2^2), \ldots, (s_1^T, s_2^T) : s_1^T = 20 \oplus s_2^T = 20, (s_1^i, s_2^i) \neq (s_1^{i-1}, s_2^{i-1})\}.$$

The transition probabilities for the current s_1, s_2 are

$$\begin{aligned}
\Pr((s_1, s_2) \to (s_1 + 1, s_2)) &= \frac{t_1 p_1}{t_2 p_2 + t_1 p_1} \\
\Pr((s_1, s_2) \to (s_1, s_2 + 1)) &= \frac{t_2 p_2}{t_2 p_2 + t_1 p_1} \\
\Pr((s_1, s_2) \to (s_1 + 1, s_2 + 1)) &= 0
\end{aligned}$$

The utility is

$$u_1((s_1, s_2)) = \begin{cases} 1 & \text{if } s_1 = 20 \\ 0 & \text{if } s_2 = 20 \end{cases}$$

Then for $z \in Z$ such that $|z| = T$, the MIVAT estimators are

$$\hat{u}_{V_1}(z) = u(z) - \sum_{i=1}^{T} \left(V_1((s_1^i, s_2^i)) - \left[\frac{t_1 p_1}{t_2 p_2 + t_1 p_1} V_1((s_1^{i-1} + 1, s_2^{i-1})) + \frac{t_2 p_2}{t_2 p_2 + t_1 p_1} V_1((s_1^{i-1}, s_2^{i-1} + 1)) \right] \right)$$

$$\hat{u}_{V_2}(z) = -\hat{u}_{V_1}(z)$$

In general, we can apply this same technique to any extensive game with discrete actions and continuous time. The time dimension can be eliminated by noting that the transitions caused by discrete actions are independent of the length of time before that transition.

For continuous actions, we can look at the case without agents: stochastic processes. Instead of a summation over rounds, we have an integral over time. The process $\{X_t\}$ starts at time 0 and

[3] This realization was obtained from work on Markov control processes [Dahl, 2001] where they suggested a similar modification for their formalism, Markov control processes.

terminates at time τ with some payoff u on the terminal history X_τ. We will look at (Itô) diffusions X_t,

$$dX_t = b(X_t)dt + \sigma(X_t)dB_t$$

where B_t is m-dimensional Brownian motion, $b : \mathbb{R}^n \to \mathbb{R}^n$ and $\sigma : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ satisfy

$$|b(x) - b(y)| + |\sigma(x) - \sigma(y)| \leq D|x - y| \;\; x, y \in \mathbb{R}^n$$

where $|\sigma|^2 = \sum |\sigma_{ij}|^2$ and $D \in \mathbb{R}$.

Theorem 7.2.4 (Theorem 7.4.1 (Dynkin's formula) [Oksendal, 1998]) *Let $f \in C_0^2(\mathbb{R}^n)$ (a continuous function) and generator A be defined as*

$$Af(x) = \sum_{i=1}^{n} b_i(x)\frac{\partial f}{\partial x_i} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\sigma\sigma^T)_{i,j}(x)\frac{\partial f}{\partial x_i \partial x_j}$$

For some finite stopping time $\tau \in \mathbb{R}$

$$E^x[f(X_\tau)] = f(x) + E^x[\int_0^\tau Af(X_t)dt]$$

From this theorem, we can see that if we set $x = X_0$, the luck term in the MIVAT estimator is

$$L_V(X_\tau) = V(X_0) - V(X_\tau) + \int_0^\tau AV(X_t)dt$$

because

$$E[L_V(X_\tau)] = V(X_0) - E^{X_0}[V(X_\tau)] + E^{X_0}\left[\int_0^\tau AV(X_t)dt\right] = 0.$$

Brownian motion is an example of a continuous time and action process. For n-dimensional Brownian motion, we have $dX_t = dB_t$ ($b = 0, \sigma = I_n$). Then, the generator of B_t with $f = f(x_1, ..., x_n) \in C_0^2(\mathbb{R}^n)$ is $Af = \frac{1}{2}\sum_{i=1}^{n}\frac{\partial f}{\partial x_i^2}$. In this case, $A = \frac{1}{2}\Delta$, where Δ is the Laplace operator. Now, for example, if an option follows Brownian motion, we can use this model to obtain a low variance prediction of $u(X_\tau)$, the value of the option at time τ from now.

In this Chapter, we have shown that extensive games encompass a broad range of domains for agent evaluation and that the MIVAT framework can be extended to encapsulate even more. In the next Chapter, we conclude our work on MIVAT, summarizing our main contributions and suggesting worthwhile directions for future work on MIVAT.

Chapter 8

Conclusion

In this section, we summarize the contributions presented in this thesis and discuss several avenues for future work in variance reduction using MIVAT.

8.1 Contributions

Our main contribution in this thesis was to unify and expand on previous work using control variates for variance reduction. We illustrated the relationship between control variates, advantage sum estimators in finite extensive games and the similar estimator framework in Markov chain processes. We then introduced MIVAT, the Informed Value Assessment Tool, a general automated approach to obtaining low-variance, unbiased estimates 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 advantage sum estimators 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 poker, six-player limit poker and two-player limit poker with imperfect information, where no previous estimators were available, based solely on simple poker features and logs of past experience. With the lower-variance estimator in two-player limit poker with limited information, the University of Alberta can further improve their expert-level two-player limit poker bot.

This work also has two other contributions: an increased theoretical understanding of value functions in advantage sum estimators and wider scope in the applicability of advantage sum estimators. We proved that using the estimator's variance as the loss function to find a value function for advantage sum estimators is optimal. We have shown that a value function satisfying the variance loss is an improvement on a value function predicting the expected utility suggested in the advantage sum work. In particular, we illustrated the superior convergence properties of minimizing the vari-

ance and showed theoretical and empirical examples where we obtain more variance reduction by minimizing variance rather than predicting the utility.

In addition to improving the practical applicability of advantage sum estimators by learning the value function, we have broadened the scope of applicable domains by incorporating continuous actions and infinite-horizons. Advantage sum estimators were defined for discrete actions; we discussed how the advantage sum framework could be applied in the continuous action case, which relies on solving an integral. We proposed certain scenarios where the integral could be computed in closed form and, for cases without a closed form solution, discussed some numerical integration techniques to approximate the unknown integral. We also illustrated how to extend advantage sum estimators to a potentially infinite action sequence, using infinite-horizon MDPs/POMDPs as a template.

8.2 Future Work

There are four main avenues for future work: (1) extending MIVAT to use other loss functions; (2) extending the MIVAT estimator to use non-linear value functions; (3) incorporating other variance reduction techniques into the MIVAT framework; and (4) implementing MIVAT for stochastic domains other than poker. This section discusses some approaches for this future work and the benefits from exploring these problems.

8.2.1 Loss Functions

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. Some robust alternatives to least squares include L-estimators, such as least absolute value regression (L_1-norm), least median of squares regression and least trimmed squares regression; R-estimators; and M-estimators, such as iteratively reweighted least squares [Andersen, 2007]. These loss functions have their own issues, such as the instability of L_1 minimization, but could prove to be an improvement in some scenarios over the L_2 loss. Moreover, though still related to minimizing the variance, the L_1 loss instead minimizes the absolute difference between the estimate and its sample average. To keep the variance term in the loss, we could look at adding penalty terms to the L_2 loss, such as an L_1 penalty. Finally, we could investigate alternative losses that might have higher variance on average but provide better estimators for typical agents.

8.2.2 Non-linear Value Functions

In this work, we have derived a closed form solution for linear value functions. Though linear value functions can be powerful, they are highly dependent on the choice of features. Optimizing non-

linear value functions may alleviate the challenge of constructing good features and add modeling power to the value function. Recall that our loss function is the variance of the estimator

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

Most non-linear machine learning techniques do not obviously apply to this loss function because they assume that the input features X are used to predict some output Y. There are two potential approaches to using non-linear techniques with the variance loss function. First, it may be possible to look at $\hat{u}_{V_j}$ as the learned function with the output variable we are trying to predict being $E[\hat{u}_{V_j}|\sigma]$ (approximated with $\frac{1}{T}\sum_{t=1}^{T} \hat{u}_{V_j}(z_t)$). If this extension is not possible, we expect that many non-linear machine learning techniques could be modified to use this different loss function. The first obvious non-linear formulation is the *dual* of this problem, which we have solved but not included in this work. More interesting extensions would involve re-deriving solutions using the variance loss function for popular non-linear machine learning techniques, such as neural networks and decision trees.

We can also use combinations of linear models to mimic non-linearity. For example, a common approach is to use random forests [Breiman, 2001]. The procedure involves splitting on a random feature with a randomly chosen value in the feature's range and repeating that process to some desired depth. Linear value functions are learned for each root of the tree (i.e., on the subset of data in that split) using the variance loss. Trees can be evaluated on hold out data and correspondingly discarded or kept in the forest. After obtaining a desired number of trees, they can be linearly weighted according to the variance loss function. With more depth and number of trees in the random forest, the linear value function becomes more and more non-linear.

8.2.3 Other Variance Reduction Techniques

We could further reduce variance with the MIVAT framework by exploring some of the other general variance reduction techniques, including common random numbers, antithetic variates, importance sampling, conditional Monte Carlo and stratified sampling. Like with control variates, we may find simplifications to these more general techniques when looking specifically at extensive games. If we explicitly illustrate or simplify the combination of MIVAT with these other variance reduction techniques, we could significantly extend the practical applicability and impact of MIVAT.

For example, the University of Alberta uses the advantage sum estimators framework with common random numbers and importance sampling to obtain low-variance estimators in two-player limit poker. They were able to reduce the number of required games from about 20,000 to about 200, enabling them to make statistically significant conclusion about performance against human players in two-player limit poker. Since DIVAT is specific to two-player limit poker, they were not able to obtain this variance reduction in multi-player limit or no-limit poker. With the MIVAT results

in this work combined with these two techniques, we might be able to bring the required number of games near 1000 for multiplayer limit or two-player no-limit poker. The variance reduction, however, would still not be enough against human players. If more of the above approaches could be incorporated, potentially the variance reduction from MIVAT would be enough to hold a competition against human experts for no-limit poker, a more widely played game.

Other techniques may also have to be explored to improve variance reduction when learning a value function with limited data. Estimating the weights for the linear value function on separate data may be difficult or impossible due to limited sampling. As mentioned, approximating the weights on the same data being evaluated may cause the estimator to be biased. We can guarantee unbiasedness if the utilities and control variates are multinormal [L'Ecuyer, 1994] (the control variates being the difference between the value before and after a chance node). It would be interesting to explore in what situations this condition holds or can be enforced. We could also explore some of the techniques for reducing bias mentioned in Section 3.2, including jackknifing and splitting [Nelson and Richards, 1990].

8.2.4 Other Domains

There are several domains in which MIVAT could improve agent evaluation, whether or not there is an explicit game formulation. An interesting application of MIVAT to a case with explicit formulations is the Annual Reinforcement Learning (RL) Competition, which uses MDPs and POMDPs. The RL Competition draws competitors from around the world and includes a broad range of domains, from standard RL benchmarking domains, like acrobot, to more complicated ones with continuous actions, like the octopus domain and helicopter hovering. Rankings are based on accumulated reward, which can have high variance; statistical significance, however, is not reported. Examining statistical significance could help improve the validity of the rankings and attract more interest to the competition.

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], the Trading Agent Competition [Wellman *et al.*, 2003], and a simulated golf task for robotics 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.

8.3 Summary

Overall, we have seen that MIVAT is a general and principled approach to automating variance reduction, able to match a state-of-the-art variance reduction tool in two-player limit poker with simple features and little design time. The significant variance reduction seen in two-player limit poker, however, may not be representative of most domains without some additions, such as adding other variance reduction techniques or extending the MIVAT estimator beyond linear value functions. The MIVAT framework provides a well-justified, general approach to simplifying the use of control variates and provides a sound baseline for future work in variance reduction for agent evaluation.

Bibliography

[Andersen, 2007] R. Andersen. *Modern methods for robust regression*. Sage Publications (CA), 2007.

[Avramidis and Wilson, 1993] A.N. Avramidis and J.R. Wilson. A splitting scheme for control variates. *Operations Research Letters*, 14(4):187–198, 1993.

[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.

[Boucheron *et al.*, 2005] Stéphane Boucheron, Olivier Bousquet, and Gabor Lugosi. Theory of classification : A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

[Bowling *et al.*, 2008] Michael Bowling, Michael Johanson, Neil Burch, and Duane Szafron. Strategy evaluation in extensive games with importance sampling. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML)*, pages 72–79, 2008.

[Breiman, 2001] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[Brown and Forsythe, 1974] Morton B. Brown and Alan B. Forsythe. Robust Tests for Equality of Variances. *Journal of the American Statistical Association*, 69:364–367, 1974.

[Crites and Barto, 1996] Robert Crites and Andrew Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*, volume 8, pages 1017–1023, 1996.

[Dahl, 2001] F.A. Dahl. Variance reduction for markov chain processes using state space evaluation for control variate. *Journal of Operations Research*, 52(12):1402–1407, 2001.

[Estelle *et al.*, 2004] Joshua Estelle, Yevgeniy Vorobeychik, Michael P. Wellman, Satinder P. Singh, Christopher Kiekintveld, and Vishal Soni. Strategic interactions in the tac 2003 supply chain tournament. In *Computers and Games*, pages 316–331, 2004.

[Kan, 2007] Morgan Kan. Postgame analysis of poker decisions. Master's thesis, University of Alberta, 2007.

[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.

[L'Ecuyer, 1994] Pierre L'Ecuyer. Efficiency improvement and variance reduction. In *WSC '94: Proceedings of the 26th conference on Winter simulation*, pages 122–132, San Diego, CA, USA, 1994. Society for Computer Simulation International.

[Lemieux and La, 2005] Christiane Lemieux and Jennie La. A study of variance reduction techniques for american option pricing. In *WSC '05: Proceedings of the 37th conference on Winter simulation*, pages 1884–1891. Winter Simulation Conference, 2005.

[Lidebrandt, 2007] Thomas Lidebrandt. Variance reduction: Three approaches to control variates. Master's thesis, Stockholm University, October 2007.

[Neal, 1993] R.M. Neal. *Probabilistic inference using Markov chain Monte Carlo methods*. 1993.

[Nelson and Richards, 1990] Barry L. Nelson and D. Richards. Control variate remedies. *Operations Research*, 38(6):974–992, 1990.

[Ng *et al.*, 2004] Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *ISER*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 363–372. Springer, 2004.

[Oksendal, 1998] B. Oksendal. *Stochastic differential equations*. Springer Berlin, 1998.

[Ortner, 2007] R. Ortner. Linear dependence of stationary distributions in ergodic Markov decision processes. *Operations Research Letters*, 35(5):619–626, 2007.

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

[Puterman, 1994] M. L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[Sodomka *et al.*, 2007] Eric Sodomka, John Collins, and Maria L. Gini. Efficient statistical methods for evaluating trading agent performance. In *AAAI*, pages 770–775, 2007.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[Tricki, 2009] Tricki. General methods for estimating integrals. World Wide Web electronic publication, 2009.

[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.

[White and Bowling, 2009] Martha White and Michael Bowling. Learning a value analysis tool for agent evaluation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1976–1981, 2009.

[Wikipedia, 2009a] Wikipedia. List of integrals of exponential functions — wikipedia, the free encyclopedia, 2009. [Online; accessed 9-November-2009].

[Wikipedia, 2009b] Wikipedia. Numerical integration — wikipedia, the free encyclopedia, 2009. [Online; accessed 6-November-2009].

[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.